Lecture Notes in
Artificial Intelligenc 869
Subseries of Lecture Notes in Computer Scien

Zbigniew W. Raś
Maria Zemankova (Eds.)

Methodologies for Intelligent Systems

8th International Symposium, ISMIS '94
Charlotte, North Carolina, USA, October 1994
Proceedings

Springer-Verlag

University of North Carolina at Charlotte
Department of Computer Science
Intelligent Systems Laboratory
Charlotte, N.C. 28223

tel: 704-547-4567

February 2, 1995

Zbigniew W. Ras
Professor of Comp. Sci.
ras@mosaic.uncc.edu

Defense Technical Information Center
Building 5, Cameron Station
Alexandria, Virginia 22314

RE: Report
Department of Navy Grant N00014-94-1-0799

Dear Sir,

Enclosed is a copy of ISMIS'94 proceedings (regular sessions) as you requested.
We appreciate very much your sponsorship.

Thank you.

Best regards,

Zbigniew W. Ras

Lecture Notes in Artificial Intelligence 869

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Zbigniew W. Raś   Maria Zemankova (Eds.)

Methodologies for Intelligent Systems

8th International Symposium, ISMIS '94
Charlotte, North Carolina, USA
October 16-19, 1994
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Jaime G. Carbonell
School of Computer Science, Carnegie Mellon University
Schenley Park, Pittsburgh, PA 15213-3890, USA

Jörg Siekmann
University of Saarland
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany

Volume Editors

Zbigniew W. Raś
Department of Computer Science, University of North Carolina
Charlotte, NC 28223, USA

Maria Zemankova
MITRE Corporation
7525 Colshire Dr., Z267, McLean, VA 22102, USA

This work relates to Department of the Navy Grant N00014-94-1-0799 issued by the Office of Naval Research. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

CR Subject Classification (1991): I.2, F.4.1

ISBN 3-540-58495-1 Springer-Verlag Berlin Heidelberg New York

CIP data applied for.

Typesetting: Camera ready by author
SPIN: 10479081 45/3140-543210 - Printed on acid-free paper

Preface

This volume contains the papers selected for presentation at the Eight International Symposium on Methodologies for Intelligent Systems - ISMIS'94, held in Charlotte, North Carolina, October 16-19, 1994. The symposium was hosted by the University of North Carolina at Charlotte and sponsored by Office of Naval Research, Oak Ridge National Laboratory, MITRE Corporation, UNC-Charlotte, and IBM-Charlotte.

ISMIS is a conference series that was started in 1986 in Knoxville, Tennessee. It has since then been held in Charlotte (North Carolina), once in Knoxville, once in Torino (Italy), and once in Trondheim (Norway).

The Program Committee selected the following major areas for ISMIS'94:

- Approximate Reasoning
- Evolutionary Computation
- Intelligent Information Systems
- Knowledge Representation and Integration
- Learning and Adaptive Systems
- Logic for Artificial Intelligence
- Methodologies (Modeling, Design, Validation)

The contributed papers were selected from more than 120 full draft papers by the following Program Committee: Luigia Carluci Aiello (Roma U., Italy), Alan Biermann (Duke U.), Piero Bonissone (GE), Jacques Calmet (U. Karlsruhe, Germany), John Campbell (U. College - London, England), Jaime Carbonell (CMU), Wesley Chu (UCLA), Misbah Deen (Keele U., UK), Robert Demolombe (CERT-Toulouse, France), Eric Dietrich (SUNY Binghamton), Jon Doyle (MIT), Douglas Fisher (Vanderbilt U.), Michael Georgeff (AAII, Australia), David Hislop (US Army Research Office), Matthias Jarke (RWTH Aachen, Germany), Leo Joskowicz (IMB - Yorktown Heights), Yves Kodratoff (U. Paris VI, France), Jan Komorowski (U. Trondheim, Norway), Kurt Konolige (SRI), Catherine Lassez (IBM Yorktown Heights), Michael Lowry (NASA Ames), Robert Meersman (Tilburg U., Netherlands), Zbigniew Michalewicz (UNC-C), Ryszard Michalski (George Mason), Jack Minker (U. Maryland), Masao Mukaidono (Meiji U., Japan), Lin Padgham (Linkoping U., Sweden), Rohit Parikh (CUNY), Zdzislaw Pawlak (Warsaw U. Tech., Poland), Francois Pin (ORNL), Henri Prade (U. Paul Sabatier, France), Luc De Raedt (Catholic U. Leuven, Belgium), Zbigniew Ras (UNC-C), Lorenza Saitta (U. Torino, Italy), Erik Sandewall (Linkoping U., Sweden), Timos Sellis (National Tech. U. Athens, Greece), Yoav Shoham (Stanford U.), Sal Stolfo (Columbia U.), Richmond Thomason (U. Pittsburgh), Ralph Wachter (ONR), Marianne Winslett (U. Illinois - Urbana), Carlo Zaniolo (UCLA), Maria Zemankova (MITRE), Jan Zytkow (Wichita State). Additionally, we acknowledge the help in reviewing the papers from: Yongyuth Aramkulchai (Imperial College, UK), Chitta Baral (U. Texas - El Paso), Christer Baeckstroem (Linkoping U., Sweden), Marianne Baudinet (U. Libre de Bruxelles, Belgium), Pierre Berlandier (INRIA, France), Sanjiv Bhatia (U. Missouri - St. Louis), Isabelle Borne (Open U., England), Patricia Carbone (MITRE), Steve Chenoweth (NCR), Jan Chomicki (Kansas State U.), Bill Chu (UNC-C), Robin Cockett (U. Calgary, Canada), Luca Console (U. Torino, Italy), Ray D'Amore (MITRE), L. Degerstedt (Linkoping U., Sweden), Luis Farinas del Cerro (U. Calabria, Italy), Bipin Desai (Concordia U., Canada), Olga De Troyer (Tilburg U., Netherlands), Rose Dieng (INRIA, France), Dimiter Driankov (Linkoping U., Sweden), Christoph Eick (U. Houston), Peter Flach (Tilburg U., Netherlands), Adam Gadomski (ENEA, Italy), Yolanda Gil (ISI/USC), John Grant (Towson State U.), Jerzy Grzymala-Busse (U. Kansas), Sablon Gunther (Catholic U. Leuven, Belgium), Tibor Gyires (Illinois State U.), Mirsad Hadzikadic (UNC-C), Lucja Iwanska (Wayne State), Cezary Janikow (U. Missouri -St. Louis), Yuejun Jiang (Imperial College, UK), Debby Keen (U. Kentucky), Mieczyslaw Kokar (Northeastern U.), Diana Komnenovic (Keele U., UK), Akhil Kumar (Cornell U.), Sukhamay Kundu (LSU), Guillaume Le Blanc (U. Paris VI, France), Jorge Lobo (U. Illinois - Chicago), Anthony Maida (U. SW Louisiana), Jacek Maitan (UNC-C), M. Matskin (Trondheim U., Norway), T.L. McCluskey (U. Huddersfield, UK), Artur Mikitiuk (U. Kentucky), Neil Murray (SUNY-Albany), Claire Nedellec (U. Paris VI, France), Ulf Nilsson (Linkoping U., Sweden), Madhura Nirkhe (Florida Atlantic U.), E.M. Oblow (ORNL), Levent Orman (Cornell U.), Luigi Palopoli (U. Calabria, Italy), Sergey Petrov (ORNL), Gregory Piatetsky-Shapiro (GTE Lab.), Jan Plaza (U. Miami), Luigi Portinale (U. Torino, Italy), Arcot Rajasekar (U. Kentucky), Satyendra Rana (Wayne State), Helena Rasiowa (U. Warsaw, Poland), Robert Reynolds (Wayne State), Celine Rouveirol (U. Paris VI, France), Pasquale Rullo (U. Calabria, Italy), Steven Salzberg (Johns Hopkins U.), Francesco Scarcello (U. Calabria, Italy), Michele Sebag (Ecole Polytech., France), Shashank Shekhar (ORNL), Andrzej Skowron (U. Warsaw, Poland), Roman Slowinski (Tech. U. Poznan, Poland), Roman Swiniarski (San Diego State U.), Bhavani Thuraisingham (MITRE), Pietro Torasso (U. Torino, Italy), C. Vasudevan (Florida Atlantic U.), Gilles Venturini (U. Paris VI, France), Hans Weingand (Tilburg U., Netherlands), S.K. Michael Wong (U. Regina, Canada), Wlodek Zadrozny (IBM - Yorktown Heights), Wojtek Ziarko (U. Regina, Canada).

The Symposium was organized by the Department of Computer Science, University of North Carolina at Charlotte. The Organizing Committee consisted of Bill Chu, Mirsad Hadzikadic, Karen Harber, Linda Kroff, Rick Lejk, M.S. Narasimha, and Zbigniew Ras.

We wish to express our thanks to William Campbell (NASA), Jaime Carbonell (CMU), Keh-Hsun Chen (UNC-C), Ed Fox (Virginia Tech.), B. Chandrasekaran (Ohio State) and Robert Meersman (Tilburg U., Netherlands) who presented the invited addresses at the symposium. We would like to express our appreciation to the sponsors of the symposium and to all who submitted papers for presentation and publication in the proceedings. Special thanks are due to Alfred Hofmann of Springer Verlag for his help and support.

Finally, we would like to thank Karolina Pauzewicz whose contribution to organizing this symposium was essential to its becoming a success.

July 1994 Z.W. Ras, M. Zemankova

Table of Contents

Invited Talks

Communications

I. Approximate Reasoning

II. Evolutionary Computation

III. **Intelligent Information Systems**

IV. **Knowledge Representation**

V. **Methodologies**

VI. **Learning and Adaptive Systems**

VII. **Logic for Artificial Intelligence**

Distributed Earth Science Validation Centers for Mission to Planet Earth

William J. Campbell
Nicholas M. Short, Jr.
Patrick Coronado
Robert F. Cromp

NASA/Goddard Space Flight Center, Code 935, Greenbelt, MD 20771

Abstract. By the end of the century, NASA will launch a series of satellites to study the Earth. NASA will store the Earth data in one of the world's largest information systems, whose size will be 1,000 times larger than the Library of Congress. Based upon a current infrastructure of Earth science validation centers throughout the world, this paper describes a highly distributed architecture for the management, processing, and quality assurance of data that complements current high-volume, reliable NASA archives.

1 Introduction

Success of recent U.S. environmental initiatives depends upon increasing the availability of data to the scientific community who will be interpreting space-based observations about ozone depletion and greenhouse effects, land vegetation and ocean productivity, and desert/vegetation patterns to name a few. Part of NASA's role in the Mission to Planet Earth (MTPE) initiative is to take a proactive leadership role in the management of space and Earth science data by making those data accessible to scientists worldwide in order to foster the new field of Earth Systems Science.

Yet, ensuring the proper processing and retrieval of these data is a formidable task because of the enormous and daunting size, geographic diversity, multidisciplinary nature and complexity of the Earth Science data systems. For example, the Earth Observing System (EOS) system will require that over one terabyte per day be collected and processed into several levels of science data products and models. Furthermore, in order to prevent a backlog of data, processing throughput must match varying collection rates so that all products can be generated and stored within several hours after observation. Over the 15-year life of EOS, this system will collect an estimated 11,000 terabytes of raw, processed, and analyzed data to be stored in the EOS Data and Information System (EOSDIS). These data will be found by searching through a meta database or "card catalogue" containing both simple search keys obtained from engineering specifications and complex keys derived from the contents or known features in the incoming sensor data. Once found, these data will be used in the analysis phase to generate special products that utilize both generic image processing techniques and domain-specific algorithms developed by scientists. Such understanding will be obtained from the integration of knowledge from the traditional Earth science disciplines and from the recent technologies that will be utilized to acquire, manage, process, and analyze this large amount of remotely sensed data.

The authors have been engaged in applied research using state-of-the-art techniques for handling the automatic archiving and querying of scientific spatial databases. As a result of over 10 years of R&D and experience in the deployment of ground stations around the world, we have developed a dynamic testbed called the Intelligent Information Fusion System (IIFS) that is being used to test, integrate and demonstrate various technologies. The motivation behind this research has been to identify, and where possible minimize or eliminate, potential problem areas facing NASA in its mission of gathering and analyzing remotely-sensed imagery in both Earth and space disciplines. The objective of this paper is to illustrate the application of information technologies to the concept of small, efficient, and self-contained validation centers having real-time direct data readout over localized regions from Earth observing satellites. These globally distributed validation centers are designed to complement central archives by providing flexibility for the development of new algorithms, the acquisition of field data, the real-time production of non-standard Earth science data products, and the generation of accurate search keys for access into the main EOS archive(s).

Section 2 describes the background on validations centers while section 3 illustrates the technical approach to current centers. Section 4 describes the application of the IIFS to any given validation center, while sections 5 and 6 discusses coordination among validation centers.

2 Validation Centers

Experience from past and current Earth observing sensors has underscored the importance of sustained and coordinated programs to verify sensor calibration and derived products. This situation will be exacerbated by more rigorous specifications on measurement accuracies that are required to address the geophysical and biological problems and an increase in data volume and downlink data rates. Two major functions encompass the notion of a validation center: compilation of local *In Situ* data[1] and verification/creation of space- and time- specific correlation algorithms. Dozier[12] states about calibration:

> There are changes in the signal, which are independent of any change in surface properties, caused by variations in the angle of illumination of the incident irradiance and angle of viewing by the sensor. The purpose of calibration or correction is to account for these effects.

Calibration of satellite sensors via remote validation centers is done by compiling *in situ* measurements (e.g., surface wind, surface pressure, ozone concentration, vegetation concentration, land use, land cover features, etc.) that

[1] direct field measurements - ground truth, atmospheric, oceanic.

directly correspond to sensor radiance measurements[2]. These measurements are then used to adjust calibration coefficients in a regression function for eventual matching of Earth features to direct observations. In addition, the ability of a validation center to acquire various satellites enables the use of these data for cross calibrating new satellites that are replacing the existing ones. This will provide data continuity and function as a baseline calibrated data set; this baseline would have been calibrated and validated with the local *in situ* measurements. Of course, validation centers must ensure accuracy by providing routine quality control of the *in situ* data.

In order for the data from these missions to be useful in quantifying long-term trends in oceanic, biological, land and atmospheric processes, comprehensive and consistent calibration and algorithm validation programs must include product and calibration comparisons when missions overlap in time. This is due to the instruments not being identical in terms of their radiometric characteristics and in terms of their derived product dependency on sensor-specific algorithms. Therefore, care must be taken to ensure that the algorithms for atmospheres, bio-optical and others, produce reasonably similar derived products. Examples of these phenomena, that cause or are due to climate global change, can be categorized into six basic groups: 1) Stratospheric events and processes, 2) Pollution events and transport, 3) Tropospheric events and processes, 4) Solar-terrestrial effects, 5) Ice systems and 6) Bio/Ecological cycle interruptions. Because of the short time spans of several of these events, achieving accurate calibration estimates requires that *in situ* data be acquired as close to an event as possible.

In addition to calibration, further considerations must be taken during the mapping of calibrated sensor values to Earth features. Dozier[12] states that:

> Significant interpretation of most remotely sensed data depends on ancillary data and knowledge of the geographic area. The more the user-interpreter knows about the area over which the data were obtained, the better the interpretation. Some ancillary data must be obtained during the acquisition of the airborne or spaceborne remote-sensor data.

With this in mind, validation centers can cut costs of *in situ* data acquisition by having local users make local measurements to take into account regional environmental variability. As these users are the most knowledgable about their respective regions, these *In Situ* data will likely be the most accurate.

Because of the ability to account for local variability, validation or real-time direct readout centers will be of great interest to the environmental research community to answer fundamental questions on Earth events (whether natural or man-made) that may impact global climate change. These time and space variant events can include mesoscale convective complex cloud systems, ocean storms and hurricanes, which often occur where conventional observations are sparse or time intervals between observations are longer than the critical time for detection of the event.

[2] i.e., satellite sensor output as a signal response in the frequency or band of interest.

Such circumstances have slowed progress in understanding these and other time variant phenomena, particularly with regard to longer temporal changes in the Earth's environmental systems, such as the impact of changes in cloudiness on radiation balance or ocean warming on hurricane frequency. Additional examples include volcanic eruptions, forest fires, snow/ice storms, Antarctic whiteouts, desert dust storms, industrial effluents, natural biota bloom, oil slicks and widespread agricultural burning. All of these are believed to have significant impact on regional acid rain problems and on the radiation balance on large hemispherical scales.

These environmental scenarios have occurred over the past several years at validation or direct readout testbed centers that have been installed at numerous sites worldwide, such as Mongolia, Guam, Fiji, Hawaii, Russia, Venezuela, Chile, Argentina and Bangladesh. These systems range in complexity and functionality depending on the countries' prime environmental concerns. For example, Mongolia, uses its validation center to monitor forest fires, snow/ice cover, vegetation coverage and meteorology. The Mongolians demonstrated the need for worldwide validation centers by finding a correlation between a grass type common to Mongolia and the vegetation index that is currently a measure of vegetation density. These correlation improvements can also be applied to other areas of the world that may grow this same type of vegetation. Unfortunately, there has been no mechanism for this knowledge transfer to occur other than mailing digital tapes and documents and there has been minimal incentive for the Mongolians to share their new correlations with the outside world. There are two major reasons why intersite cooperation has been minimal. First, the user community is unaware of the other algorithm/correlation developments, and, second, an increased support level will be required for validation centers to provide new or improved algorithms, causing an increase in costs. Hence, a mechanism within the system will be needed in order to narrow the communication gap; this issue will be addressed later in this paper.

The Guam, Fiji and Bangladesh systems focus on the atmospheric aspect of the environment. Their prime areas of interest include cyclones (typhoons), tropical storms, inversion systems and precipitation. From these interests, specialized algorithms were developed to study environmental conditions. Such algorithms include the Dvorak Classification (to measure typhoon strength), Log 10 Spiral (to locate the eye of a storm), and storm history archive (to predict storm trajectory), to mention only a few.

Because real-time data acquisition makes early warning of catastrophes possible, these systems, over the past six years, have helped save an estimated 1,000,000 people from natural catastrophes. Examples of this include Bangladesh where two weeks after the installation of a similar system, the system discovered a typhoon 500 miles from the coast. This early warning allowed the movement of over 200,000 people to higher elevation thereby avoiding certain drowning due to flooding. In Mongolia, the TIROS[3] ground system detected a blizzard coming from Siberia that eventually covered half of Mongolia. With this information,

[3] Television Infrared Observing Satellite

the Mongolians were able to prepare for the storm, saving thousands of lives as well as livestock. When the blizzard subsided, it left behind four ft. of snow in a 300 Km^2 area. Because of the many examples of this magnitude that have occurred over the last few years, these testbed validation centers have provided evidence of their importance by not only providing global change and disaster monitoring but also by supporting the production of calibration and validation information for existing and future Earth observing sensors.

3 Validation Center-System Description

The design, development, and deployment of these systems have provided unique and invaluable information for the architecture of the next generation validation centers. These sites have become the testbed for many of the technologies described in this paper, thereby, giving the developers an insight into the technical feasibility of inexpensive ground systems. Before discussing the next generation, this section will illustrate a typical validation center scenario.

The validation center operation begins with the acquisition of an RF signal from the transmitting satellite. This signal is normally captured by a variable size parabolic dish, depending on frequency and data rates. The signal is then downconverted to a lower frequency for purposes of transportation via cable to a demodulator (receiver). Here the signal carrier is stripped leaving the raw data (video) which is then sent to a bit synchronizer. At this point the digital data embedded in analog form is converted to a digital bit stream whereby the bit synchronizer creates clock pulses that correspond to the data. This information, along with the conditioned digital bit stream is passed on to a frame formatter. The data is converted to computer readable words of variable length, depending on pixel resolution. The computer readable data now passes through the bus to a storage media such as mass storage or a hard disk. Once the full downlink is completed, the data, called level-0 or raw data, is ready for image processing.

Even before acquisition occurs, an important validation center operation, planning and scheduling, must take place. A plan must exist to determine which satellites are to be tracked and acquired. This decision is based on the user's requested products and the resource availability. Once these satellites are selected, different orbital models are run depending on the satellite type. After these models execute, pass selection occurs and processing tasks are placed in an ingest schedule. Some of these functions are automated using a very rudimentary constraint-based predecessor of the planning and scheduling subsystem described later in this paper.

Once the preliminary planning and scheduling is completed, data acquisition takes place and the raw data is placed in a temporary disk file to await processing. The data generally has standard preprocessing procedures, such as deinterleaving of channel[4] data, stripping of the auxiliary information, georegistration, sectorization, warping and projection. These processes, although straightforward, create considerable operational problems, since the stations can receive

[4] where a channel is a sample of the EM spectrum.

data from three to six satellites at a time and each of the downlink data sets has to go through its corresponding aforementioned procedures. Additionally, there is generally more than one way to perform the same processes. For example, georegistration can be accomplished using every pixel point on the image or can be subsampled to a range of 30 to 60 pixels; the higher the subsample size the less accurate the georegistration. Given this situation, the obvious choice would be to use the most accurate algorithm. Unfortunately, this choice can create logistical difficulties that result in increased utilization of the CPU compared to a 40 pixel algorithm. Subsequently, unless the user has sufficient computational resources, he/she must make sacrifices in accuracy, such as increasing the pixel subsampling in order to allow complete processing of the data within an acceptable time frame. These accuracy problems also occur for the other preprocessing procedures mentioned above. The past solution to these problems has been to rigidly design the system to generate data products in a default mode using preselected algorithms that allow for the creation of products within a reasonable amount of time. Consequently, this older design provides little flexibility in the user's choice of procedures and/or algorithms. These problems have been taken into consideration in the planning and scheduling subsystem described in this paper.

Once the preprocessing procedures have been completed, the data subsets are ready for conversion to useful image and/or graphical products. These products, which include level-2 (derived products) and level-3 (gridded to a coordinate system and averaged derived data), are then analyzed and archived onto digital tapes (4/8 mm tapes). The system keeps track of the raw data and products by placing them into pseudo bins such as tree directory structures. These structures then are linked with logical system pointers which tie all data and processes together. This mechanism has proven to be effective for low data volumes where one to three satellites are being tracked. As new algorithms, procedures, correlations and products, occur along with an increase in volume, it is essential that a scientific spatial data management system such as the IIFS be used.

4 The Intelligent Information Fusion System

We have explored many technologies over the past ten years [3, 21, 9, 8] in attempting to achieve the goal of developing the end-to-end prototype IIFS system. Any successful scientific data management system will incorporate a robust, flexible query language, an intuitive graphical (or perhaps multimedia) user interface, efficient ways of organizing the data with respect to how the end-user needs to traverse it, and a multidisciplinary view from the perspective of the science from which the data is derived. For a project with sizable holdings such as EOSDIS, the scientific data management system must allow for data holdings and computational resources to be geographically distributed, underlining the need for software that can efficiently direct the satisfying of a user's query while balancing the demands placed on the limited number and capacity of computing machines, networks, and mass storage devices.

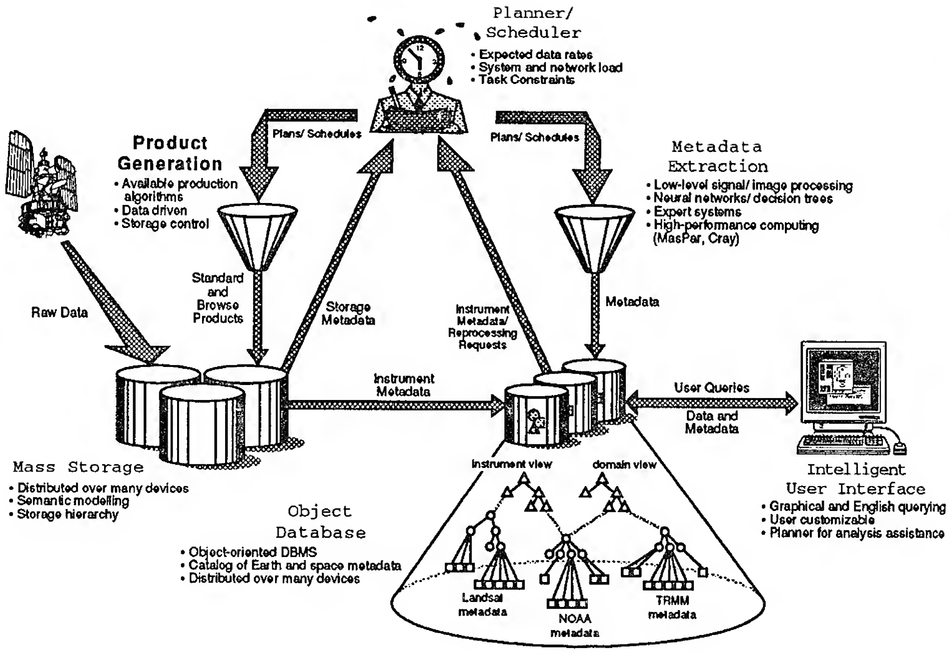


Fig. 1. The Intelligent Information Fusion System components

Since 1989, the IIFS has been based on an object-oriented database which is used to store metadata about remotely-sensed images[8, 11]. The metadata itself is organized to enable fast, efficient access to the appropriate images with respect to typical image header data, plus it incorporates innovative spatial data structures known as sphere quadtrees (SQT) that more naturally represent data acquired globally. SQTs eliminate many of the problems that conventional regularly gridded schemes exhibit when data with a spherical geometry is forced into them [15, 16, 14, 10]. Most databases for spherically distributed (e.g., astronomical, cartographic, climatological) data are not structured in a manner consistent with their geometry. As a result, such databases possess undesirable artifacts, including the introduction of "tears" in the data when they are mapped onto a flat file system. Furthermore, it is difficult to make queries about the topological relationship among the data components (e.g., adjacency of connected regions) using simple and purely symbolic means, that is, without performing floating-point arithmetic.

Additionally, we have developed a number of fast techniques for automatically extracting information about image content, enabling users to query for pertinent data sets based off of the features of scientific interest within the images themselves, thus for the first time ever enabling the formulation and satisfying of queries involving change detection and analogy, and requiring the system to have semantic knowledge of the scientific domain itself, instead of cursory syntactic understanding of nondescript symbolic header data[4, 2, 7]. Several of these classification techniques have included backpropogation neural networks[2], extended probablistic networks[26, 5], post-probablistic adjustment to classifications[7], and hybrid, expert system/neural net classifiers[20].

Characterizing the data and extracting its content has implications in the organization of the object-oriented database as well as in the efficiency and the speed of data retrieval in response to a user query. For such a system to fully answer the needs of multiple users with diverse interests, the data must be organized to permit retrieval in multiple ways. For this reason, we include in our prototype a large toolbox of techniques for characterizing and classifying the data with different levels of understanding of its content. This design will also permit us to add or suppress techniques as the prototype evolves.

Two core technologies for managing production processing and data retrieval/analysis are planning, which refers to decisions about what operations must be taken to achieve a desired goal(s), and scheduling, which refers to when these planned operations must take place [8, 22, 1, 18]. Generating plans for the analysis of remote sensing data involves complex dependencies on the location, time, atmospheric and sun angle properties, and platform involved in a given image. These factors among others, can affect how an image is best processed (e.g., to remove sensor noise or artifacts, or to generate some kind of normalized product). Moreover, the success of later processing steps depends on what earlier processing was done. All of these decisions depend on the trade-offs between time and accuracy of the image processing algorithm. When these analyses have to be scheduled on any of several hosts in competition with other analyses, the problem becomes even more complex.

The planner/scheduler must balance the trade-off between spending too little or too much time collecting statistics about processor loads, available storage space, and network communication rates and reliability. Likewise, the planner/scheduler must balance the processing loads to prevent any one processor from being a bottleneck, threatening the capability of the system to keep up with the high data rates. After retrieving data by using browse products generated by a planner/scheduler in the ingest phase, scientists currently must spend much of their time preparing their own specialized data products. With this system, the scientists themselves can use planning for data analysis/preparation in the same manner as it is used in the preprocessing phase, except for the scientists, real-time processing is not as critical[18].

Developed in cooperation with Honeywell Technology Center, the current ingest planner in the IIFS, called PlaSTiC, performs hierarchical planning and constraint-based scheduling in a resource-limited environment[1]. In PlaSTiC, tasks compete for such resources as processing time, disk space, and the use of archive servers to retrieve data from long-term mass storage. Likewise, we are currently working with NASA-ARC in adapting the COLLAGE planner[17] for the planning of post processing, retrieval and analysis tasks[18].

5 Issues in Distributed Validation Centers

The current notion of a validation system implies that the system is a stand-alone, small-scale system for performing end-to-end ground processing. With advances in networking technology and distributed computing, these small ground systems can feasibly be linked together to get complete global coverage and, hence, can provide an emergent behavior similar to centralized systems.

While the advantages of a decentralized system are obvious, several technical challenges exist. To date, most distributed systems have dealt with issues in load balancing, process(or) coordination, coherence of information, managing network bandwidth and I/O bottlenecks, graceful degradation of the processing environment when machines fail, fault diagnosis, and interoperability. Although these issues are necessary for building the "information superhighway" for bringing validation centers on-line, they do not address the higher level properties of communities of scientists cooperating together in a "virtual laboratory" [27] for remote sensing.

The AI field began addressing distributed computing when the field noticed that singular expert systems were too tightly scoped for useful problem solving. With systems emerging from the blackboard architecture[13], distributed expert systems allowed for comprehensive problems solving, where individual knowledge bases could be partitioned and maintained independently by different domain experts. Early DAI systems, such as Contract Net[5] and Actors/Open Systems, developed a protocol for negotiation and cooperation among multiple problem solving systems or "agents" [24].

These systems differ from standard client/server paradigms because each agent can perform different roles as well as many higher-level reasoning functions. For example, Contract Net was an early DAI protocol for distributed task allocation and worked by partitioning the agents into two sets: managers and contractors. Manager agents broadcast requests for contractors to work on a specific task(s). In each task announcement, information and constraints were sent to contractors who, based upon their ability to satisfy the task's constraints, bid for the right to perform the task.

In the remote sensing arena, the agent paradigm is already being proposed as an extension for distributed data retrieval [27]. For validation centers, data retrieval agents are necessary, but are not sufficient for managing other functions such as product generation. For example, while the primary purpose of validation centers is to perform sensor calibration, a secondary purpose is to test new algorithms developed within each center. Since the knowledge of each validation center will be different from others (i.e., heterogeneous), agents must search the collective for operators/algorithms that satisfy their product generation goals. Because validation centers exist in different countries which may not necessarily be willing to share access to data and processing, the distributed validation system architecture must incorporate a notion of "incentive" to persuade or

[5] Ironically, Contract Net was originally proposed for coordination of distributed sensors.

facilitate cooperation.

DAI researchers have proposed a "marketlike organization" [19, 28] for task allocation. Contract Net, for instance, allowed for competitive bidding by contractors for tasks where cost is associated as a constraint with each task announcement. For validation centers, each center could, for example, barter with others by using their own special algorithms, data, and computational resources as a common exchange "currency". A scientist in the Guam validation center, who is developing a new classification algorithm to be tested on South Pacific data, can test its generality by applying it to data or *in situ* measurements acquired through bartering with another validation system in North America, for example. Likewise, a scientist in North America could give the Guam scientist data in exchange for preliminary access to results from and potential use of the new classification algorithm. Either way, progress is made towards both removing ad hoc techniques and verification through the repetition of experimentation.

Admittedly, as most economists have realized, bartering systems are too cumbersome for practical, automated approaches because too much time is spent during negotiation and too much state information must be maintained about the availability of objects for exchange. As most modern economies have done, the abstraction of "money" can be introduced into the information economy of the distributed validation centers[28]. Of course, protection measures must exist to ensure that this virtual money actually represents algorithmic, data, and processing power "goods". Nevertheless, those scientists who produce quality algorithms will be rewarded by more virtual money, which, in turn, will allow them to test their theories on more data.

6 Extending the Current IIFS for DAI

Extending the current IIFS to execute a form of DAI can be trivially done by modifying PlaSTiC as was done in [25] or by adding both a blackboard and negotiation module using Contract Net[23]. For the latter, assuming a hierarchical organization of agents, subgoals could easily be converted to task announcements which pass the necessary state information and time-dependent constraints from the scheduler to another contractor PlaSTiC Planner, which could report back to the manager about satisfiability of the subgoal via a valid execution. This is essentially a distributed planning approach as in [6] where communication is incorporated into the planning process. Within the current validation structure, this could be easily added because NASA can currently impose standards and can avoid a totally decentralized systems that would require a notion of a "market economy".

7 Conclusion

While the primary purpose of validation centers has been to provide reliable *in situ* data, improvements in inexpensive hardware, storage technology, and intelligent software will expand the role of future centers. The IIFS architecture will allow validation centers to cost-effectively capture and encode local interpreter's knowledge needed for proper classification. Likewise, techniques from the DAI field will provide the requisite incentive necessary for cooperation and coordination among heterogeneous centers. Through this network, these centers will provide EOSDIS with inexpensive, regional knowledge necessary to validate global-scale data.

References

1. M. Boddy, J. White, R. Goldman, and N. Short, Jr. Planning applications in image analysis. In *1994 NASA-GSFC Proceedings of Space Applications of AI*, Greenbelt, MD, May 1994.
2. W. J. Campbell, S. E. Hill, and R. F. Cromp. Automatic labeling and characterization of objects using artificial neural networks. *Telematics and Informatics*, 6(3–4):259–271, 1989.
3. W. J. Campbell and L. H. Roelofs. Artificial intelligence applications concepts for the remote sensing and earth science community. In *Proceedings of the 9th Pecora Symposium*, Sioux Falls, SD, October 1984. IEEE CH2079/84/0000/232.
4. W. J. Campbell, L. H. Roelofs, and M. Goldberg. Automated cataloging and characterization of space-derived data. *Telematics and Informatics*, 5(3):279–288, 1988.
5. S. R. Chettri and R. F. Cromp. The probabilistic neural network architecture for high-speed classification of remotely sensed imagery. To appear in *Telematics and Informatics*, 10(3), 1993.
6. D. Corkill. Hierarchical planning in a distributed environment. In *Proceedings of the 1979 International Joint Conference on Artificial Intelligence*, pages 168 – 175, 1979.
7. R. F. Cromp. Automated extraction of metadata from remotely sensed satellite imagery. In *Technical Papers, 1991 ACSM-ASPRS Annual Convention*, volume 3, pages 91–101. ASCM/ASPRS, 1991.
8. R. F. Cromp, W. J. Campbell, and N. M. Short, Jr. An intelligent information fusion system for handling the archiving and querying of terabyte–sized spatial databases. In *International Space Year Conference on Earth and Space Science Information Systems.* American Institute of Physics, 1992.
9. R. F. Cromp and S. Crook. An intelligent user interface for browsing satellite data catalogs. *Telematics and Informatics*, 6(3/4):259–271, 1989.
10. R. F. Cromp and E. Dorfman. A spatial data handling system for retrieval of images by unrestricted regions of user interest. *Telematics and Informatics*, 9(3/4):221–241, 1992.
11. E. Dorfman. Architecture of a large object-oriented database for remotely sensed data. In *1991 ACSM-ASPRS Annual Convention Proceedings*, pages 129 – 143, 1991. Volume 3.
12. J. Dozier and A. Strahler. *Ground Investigations in Support of Remote Sensing*, volume 1, chapter 23. American Society of Photogrammetry, 2nd edition, 1983.
13. L. Erman, F. Hayes-Roth, V. Lesser, and Raj Reddy. The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213 – 253, 1980.
14. G. Fekete. Rendering and managing spherical data with sphere quadtrees. In *Proceedings of the First IEEE Conference on Visualization*, San Franciso, California, October 1990.

15. G. Fekete and L.S. Davis. Property spheres: A new representation for 3-d object recognition. In *Proceedings of the Workshop on Computer Vision: Representation and Control*, Annapolis, Maryland, 1984.
16. G. Fekete and L. Treinish. Sphere quadtrees: a new data structure to support the visualization of spherically distributed data. In *Proceedings of the SPIE Symposium on Extracting Meaning from Complex Data: Processing, Display, Interaction*, Santa Clara, California, February 1990.
17. A. Lansky. Localized planning with diversified plan construction methods. Publication FIA-93-17, NASA Ames Research Center, Moffett Field, CA, June 1993.
18. A. Lansky. A data analysis assistant. In *Proceedings of the 1994 AAAI Spring Symposium on Software Agents*, Stanford, CA, 1994.
19. T. Malone. Modeling coordination in organizations and markets. In A. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 151 – 158. Morgan Kaufmann, Inc., 1988.
20. N. Short, Jr. A real-time expert system and neural network for the classification of remotely sensed data. In *Technical Papers, 1991 ACSM-ASPRS Annual Convention*, volume 3. ASCM/ASPRS, 1991.
21. N. Short, Jr., W. J. Campbell, L. H. Roelofs, and S. L. Wattawa. The crustal dynamics intelligent user interface anthology. Technical Report TM–100693, NASA/GSFC, Greenbelt, MD, October 1987.
22. N. M. Short, Jr. and L. Dickens. Automatic generation of products for terabyte-size geographic information systems using planning and scheduling. *accepted for publication in the International Journal of Geographic Information Systems*, 1994.
23. E. Simoudis and M. Adler. Integrating distributed expertise. *International Journal of Intelligent and Cooperative Information Systems*, 1(3 & 4):393 – 410, 1992.
24. R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In A. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 357 – 366. Morgan Kaufmann, Inc., 1988.
25. R. Smith. Multistage negotiation in distributed planning. In A. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 367 – 384. Morgan Kaufmann, Inc., 1988.
26. D. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.
27. C. Toomey, E. Simoudis, R. Johnson, and W. Mark. Software agents for the dissemination of remote terrestrial sensing data. to be published in the 3rd International Symposium on AI, Robotics, and Automation for Space, JPL, Pasadena, CA.
28. M. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *JAIR*, 1:1–23, 1993.

Causal understanding in reasoning about the world

B. Chandrasekaran

Laboratory for AI Research
The Ohio State University
Columbus, OH 43210, USA

Abstract. In this paper I survey over a decade of work on how we understand how things work. Much of this work has been conducted in the context of reasoning about functions of devices. I briefly overview a representational framework called Functional Representation, and indicate how a device representation in this language can be used for simulation, diagnosis and design. I make remarks about the generality of this approach in terms of causal understanding.

1. The Problem: Understanding how things work

For a number of years several associates and I have been investigating what it means to understand "how things work." Understanding how things work — from "how clouds make rain," through "how cancer is formed," to "how nuclear plants generate power" — is a ubiquitous cognitive activity. Given some description of physical configuration, i.e., some description of components and how they are connected, and given some knowledge of the domain of interest —the underlying laws, the kinds of behaviors the components are capable of, the kinds of causality pathways that the connections provide, etc. — we can often predict aspects of the behavior of the physical configuration as a whole, and conversely, if we are given a behavior of the configuration, we can often construct an account of how that behavior comes about. Everyday reasoning as well as specialized scientific and engineering reasoning is full of examples where we need to put together a causal explanatory account of how some behavior of interest is produced. Such an account would explain the observed behavior by appealing to our understanding of the components, how they are connected, and the underlying scientific laws. This account would constitute our understanding of how the behavior was produced. Normally we expect a certain range of problem solving and inferential capabilities of someone who claims to understand how something works. We would expect that person to answer questions about changes in behavior as the structure of the configuration changes, to be able to identify the component responsible if the behavior of the configuration is not as expected, and so on.

The first research issue is the logical structure of this kind of causal explanation and the vocabulary of explanatory terms used in it. Once we have identified the terms used in the explanation and how the explanation is organized, we have the basis for a knowledge representation language. For a given configuration and behaviors of interest, using this language, we can represent how the configuration produces the behavior. We can process this representation with appropriate inference machinery to answer the kinds of questions that we would expect a human understander to be able to answer. Developing a formal representation language to capture one's understanding of how something works, and techniques for using such representations for problem solving activities such as simulation, diagnosis and design have also been major concerns of our research.

An important underlying idea is that causal understanding of the world doesn't come simply in the form of "facts" about the world -- propositions or causal rules -- but in the form of *causal packages* which are organized from specific perspectives and which point to other causal packages. The work reviewed is based on the idea that these packages are basic units of comprehension and prediction. The Functional Representation (FR) theory is a proposal about the logical structure, representation and use of such causal packages. My associates and I have reported extensively on this research. The citations [1-3], [9-10], and [12-28] in the bibliography are an illustrative, but not a complete, listing. A review article [4] summarizes the research in some detail. For a detailed discussion of related work in the literature, the reader is referred to [4] as well. In this paper, I simply outline the approach and some of the issues, to provide a sense of the problem and our framework for it.

2. Functional Representation

Let us look at the examples that I started the paper with: "how clouds make rain," "How cancer is formed," and "how nuclear plants generate power." In this list, only the third is a *device*, that is, an artifact that is constructed such that it achieves an intended *function*. Clouds were not designed so that rain is produced (at least not in the scientific account of the world), nor is cancer an intended function of the human physiological system. Nevertheless, all of them have a structure of interconnection of physical components, and the explanation of the behavior of interest follows similar principles: we explain the behavior as arising out of the causal properties of the components, the physical laws and the interconnections. Much of the theory of causal explanation and understanding is thus applicable not only to designed artifacts, but to any configuration of causally efficacious components linked together in ways that they can affect each other. However, in much of what follows, we will talk of devices, in order to focus the discussion.

The Functional Representation framework is a proposal about the top-down representation for goal-directed, flexible reasoning that bridges abstraction levels. It was originally proposed in [19] for the causal processes that culminate in the achievement of device functions. (Some devices achieve their functions by means of causal *processes*, while the function of others is explained directly from their structure. We discuss this distinction later, but for now consider only devices wherein causal processes are the means for achieving the functions.) In FR, the function of the overall device is described first and the behavior of each component is described in terms of how it contributes to the function.

2.1. Informal overview

FR for a device has three parts:

- a description of intended *function* (independent of how the function is accomplished),
- a description of the *structure* of the device, i.e., what the parts are and how they are connected to the degree of detail chosen by the describer, and
- a description of *how* the device achieves the function, specifically a *process* description.

The part of FR that describes the function treats the device as a blackbox, i.e., it makes no assumptions about its internal structure, not to mention any processes that take place in the device. This is because the same function may be achieved in different ways, and thus a description of the function itself should not make any commitments about the structure. This may be called the "No structure in function" principle, a kind of converse of the "No function in structure" principle that is due to de Kleer. Of course for describing a function we need to describe a certain minimum amount of structure: how the device is to be embedded in the environment (e.g., where it is to be placed and how it to be connected), where on the device the operations to initiate the function are to be performed (e.g., turning the switch on for an electrical lighting circuit or an electric iron), and where the functional output is to be realized (e.g., light near the bulb, heat at the ironing plate). Other than this, we just need to describe under what conditions what kinds of predicates are to be satisfied by the device in order for it to achieve the function of interest. That is, a function is represented by describing the context of its application, the initiating conditions and the predicates that the device has to satisfy for it to achieve the function.

Of course FR as a whole enables us to combine this description of function with how the specific device achieves it, by adding descriptions of structure and the causal processes which make the function happen in device. This independence of the *what* of the function from its *how* is important to emphasize since this a point that is widely understood to be an important desideratum for representing function (e.g., [5]); FR maintains this distinction.

Representing the structure is straightforward: we simply list the component names and their functions and indicate how the components are put together to make the device, i.e., describe the relations between the components. The components functions are described using the same ideas as in the description of the device function. .

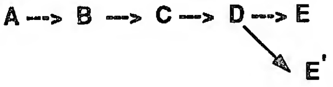
The basic idea in describing how the device achieves the function is that we need to construct a *causal process description* (CPD) of a certain type. That is, we describe how the device goes through a causal state transition process in which the initial state is the state of the device just at the time the device "starts," the final state is the state at which the device is achieving the function for which it is designed, and each state transition is explained by appealing to knowledge about components or the domain. The idea that a behavioral description of the device is the link between structure and function has been stated often in the AI literature, but our proposal on CPD takes a very specific stance about what kinds of behavioral descriptions have the explanatory power needed.

A CPD can be thought of as a directed graph whose nodes are predicates about the states of the device, and links indicate causal transitions. The links have one or more annotations that explain the transition. The annotations, according to the theory, belong to certain specific types.

Consider a simple electrical circuit with a switch, a voltage source and a resistor. Let us say that the function we are interested in is production of heat. A causal account of how this function comes about might be given as follows:

A ---> B ---> C ---> D ---> E

where A is 'switch is on,' B is 'voltage applied between terminals,' C is 'current to flow in the circuit,' D is 'current flows through the resistor,' and E is 'heat generated.' Normally "→" can be interpreted as "causes." The CPD is not a neutral description, but one that is oriented towards explaining a selected function. If the function of interest is the production of light and the resistor was a filament inside a light bulb, the link D → E, might read as follows: D is 'current flows through the resistor,' and E is 'light generated.' Of course the heat through the resistor also produces heat. If we are interested in explaining the production both of light and heat, the graph might look as follows:



where E corresponds to production of light and E' to production of heat.

Now let us examine how the transitions might be explained (i.e., annotated). "A → B" : "Closing the switch causes voltage to be applied between the terminals of the circuit." In explaining this transition we use knowledge about "closing" function of switch (the component "switch" has two functions, "open" and "close") and the "voltage" function of "battery."

The transition "B → C," "the application of voltage causes a current to flow through the circuit," invokes the connectivity-providing functions of the various connectors, and also uses a domain law, viz., Ohm's Law. Checking the connectivity functions of the connectors ensures that the structure is in fact a circuit. Ohm's Law can be used to provide qualitative or quantitative information about the quantities involved.

The transition "C → D" asserts that "current flowing through the circuit means that there is current through the resistor." This particular transition is not normally viewed as a causal transition. The explainer is simply inferring a predicate of interest, in this case, that there is current through the resistor. The explanation appeals to the meaning of a circuit. If we want only causal links, we can collapse the two links into one, "B → D," asserting, "Application of voltage causes current through the resistor." The explanation would still appeal to the connectivity functions of the connectors and to Ohm's law.

The transition "D → E," asserting that "current through the resistor causes heat to be produced," can be explained in different ways. One possibility is to simply point to the scientific laws that relate electricity to heat production in a resistor, similar to the way Ohm's Law was used earlier. Such a law can be given in a numerical or qualitative form, as needed. In this case, the link would be annotated as "By domain-law (current-to-heat equation)." In this explanation, the domain law is taken as a primitive explanation, satisfactory for the current problem solving goals.

The same transition can be explained by appealing to another process: how electricity gets converted to heat. The link would have the annotation "By-CPD(electricity-to-heat)." Someone who already understands this process can use it if needed and build the more detailed (and longer) causal story. This causal process can be explained separately for some one who doesn't already understand it.

To summarize the various explanatory annotations:

i. By-CPD: This points to another CPD that provides further details of the transition. The details of the CPD may not matter for the current purpose. If they matter, the CPD may be part of the prior knowledge of the agent, or can be explained separately. Potentially long process explanations can thus be hierarchically composed out of other process explanations, making explanation at each level shorter. CPD's can be reused. Human expertise in a domain contains knowledge of a large number of such causal processes that can be parametrized and re-used.

ii. By-Function-Of-<component> : This annotation appeals to the function of a component as the causal explanation of the transition. A major goal of causal explanation in devices is to explain the behavior of the device in terms of the properties of the components and their inter-connections. Again, a large part of expertise of human experts in a domain is in the form of knowledge about generic components and their functions (though, in many cases, how the component functions may not be known). The ability to explain the device functions partly in terms of component functions, and to explain component functions in turn in terms of the functions of its subcomponents helps in the formation of functional/component hierarchies in explanation and design. Also, components with different internal structure but the same function can be substituted.

iii. `By-Domain-Law` <law>. Another form of explanation is by appeal to domain laws. In the domain of engineering, scientific laws are the ultimate basis of explaining why the device behaves as it does.

Non-causal links: Sometimes additional, noncausal, links may need to be added to arrive at the predicate of interest. For example, for an amplifier, we may have constructed the CPD,

Voltage 1 at the input →. . . . . . .→ *Voltage 10 at the output,*

but the function that needs to be explained might be "*Amplification of 10.*" A non-causal, definitional/abstraction link can be used to arrive at the node "*amplification of 10*" from "*Voltage 10 at the output.*" Such links can be used to indicate an inference that follows from predicates in the earlier nodes.

Qualifiers: In addition to the explanatory annotations, the links may have *qualifiers* which state conditions under which the transition will take place. In FR the qualifier `Provided(p)` is used to indicate that condition `p` should hold during the causal transition for the transition to be initiated and completed, and `If(p)` to indicate that the condition `p` should hold at the moment when the causal transition is to start. The conditions can refer to the states of any of the components or substances. Many of these qualifiers are eventually translated as conditions on the structural parameters.

2.2. Structure of a device

The structure of a device is a specification of the set of components that constitute the device and the relations between the components. The components are represented by their names and by the names of their functions, which are all domain-specific strings. Components have *ports* at which they come together with other components in certain relations. The relational vocabulary can also include unintended relations, e.g., *electrical leakage between electrical components.* The components can be in unintended relations to each other as a result of malfunctions.

2.2. States and Partial States

A device *state* is represented as a set of state variables $\{V_S\}$ consisting of values of all the variables of interest in the description of the device. In describing functions and causal processes, we generally talk in terms of *partial states* of the device. A partial state is given by the values (or some constraints on the values) of a subset of state variables. In the circuit example, the nodes in the graph are partial states of the circuit. The language in which states are represented is itself not part of the representational repertoire of FR and is largely domain-specific. In economics, the state variables would be entities such as *GNP*, *inflation-rate*, etc.; in nuclear-plants, an entity might be *radiation-level.* Goel [9] has defined a state description language which is useful in describing devices that deal with material substances that change locations, e.g., those in which *substance flow* is a useful notion.

State Abstractions. Consider a device which, at one level of description of states, has one of its state variables, say s, going through the following partial states repetitively: {-1, 0, 1}. That is, the state variable is oscillating around 0. Suppose we define another state variable *?oscillating* by a process of abstraction from the values of s over time. This would be a binary variable taking on the value *Yes* if the values of s are cycling through {-1, 0, 1} and *No* otherwise. Keuneke and Allemang [15] discuss a number of issues in creating such abstractions.

2. 3. Causal Process Description

Formally, the causal process description (CPD) (see [12]) is a directed graph with two distinguished nodes, N_{init}, and N_{fin}. Each node in the graph represents a partial state of the device. N_{init} corresponds to the partial state of the device when the conditions for the function are initiated (such as turning a switch on). N_{fin} corresponds to the state where the function is achieved. Each link represents a causal connection between nodes. One or more qualifiers are attached to the links to indicate the conditions under which the transition will take place, and one or more annotations can be attached to indicate the *type* of causal explanation to be given for the transition. The graph may be cyclic, but there must be a directed path from N_{init} to N_{fin}.

2. 4. Types of Functions

Keuneke [16] has identified four types of functions: `ToMake`, `ToMaintain`, `ToPrevent`, and `ToControl`. Formal definitions of these function types have been developed [12], but for our current purposes the following informal ones should suffice. All the function types above except `ToControl` take as argument a predicate, say P_F, defined over the state variables of the device. The function is of the `ToMake` type if the goal is to make the device reach a state in which P_F is true, and after that state is reached no specific effort is needed to keep the predicate's value to True, or it doesn't matter what state the device goes to after the desired state is reached. A function is of type `ToMaintain` if the intention is to take the device to the desired state and the device has to causally ensure that the predicate remains True in the presence of any external or internal disturbance which might tend to change the device state. The function type is `ToPrevent` if the goal is to keep P_F from ever being true, and some active causal process in the device is required tc ensure it. The function type `ToControl` takes as argument a specified relation $v_0 = f(v_1,...v_n)$ between state variables v_0, v_1,...v_n, and the intent is to maintain this relationship. That is, we wish to control the values of specific variables as a function of the values of some other variables.

Passive Functions So far, all of our discussion of function has been in the context of temporally evolving causal processes. Keuneke [16] made a distinction between "active" and "passive" functions. A chair satisfies the function of providing a seat for a person. The function is normally explained as a match between the structural properties of the chair and how that meets the need for providing a seat, rather than by providing a description of a causal process. Such devices can still have parts and parts may have functions as well, and the device function as a whole will still arise from the parts achieving their functions. Each of the component functions can be defined in terms of certain structural properties.

Content-theory of functions. In specific domains, we can develop theories of elementary functions that can be combined to make more complex functions. When the domain involved is of great generality, such as mechanical force transmission, such a content theory can be widely useful. Hodges [11] has developed a set of useful basic functions in the domain of physical objects interacting by means of their shapes. Examples of such elementary functions are: *linkage, lever, gear, pulley, screw, spring,* and *container.* These functions themselves are defined in terms of a vocabulary of state change operations, such as *move, constrain, transform,* and *store.*

2. 5. Remarks on FR.

1. *The level of abstraction of the predicates in the CPD is at the level needed for the causal story.* Some of them may directly correspond to component-level state variables, while others may have only a device-level existence.
2. *FR is underdetermined by the underlying structural description.* What aspects are chosen and what levels of abstraction they are represented in depend on the problem solving goal.

3. Uses of Functional Representation

What sort of things can we do with an FR of a device? Well, it turns out that one can use the FR for the sort of things that one would expect a human understander to be able to do. Over the years, we have investigated a number of tasks.

3.1. Simulation

Qualitative simulation [6-8] is now a well-known family of techniques in AI. In this approach the behavior of the device is composed out of the behavioral description of the components, processes or state variable relations. These techniques need to be complemented by other techniques to provide the following additional capabilities that are often needed:

1. *Device-level Vs component-level abstractions.* Suppose we have an electronic amplifier, and the device's structure is described in terms of the

components: the transistors, resistors, capacitors, etc. Let us say that each of these component behaviors is described in terms of their currents and voltages. Qualitative simulation techniques would then produce a description of device behavior in terms of currents and voltages. However, the behavior of interest of the device as a whole is as an amplifier, a higher level of description. We need techniques by which the device-level behavioral abstractions are related to descriptions at the component level.

2. *Concern only with aspects relevant to the goal.* The simulation techniques of qualitative reasoning produce the values of *all* the component-level state variables which are part of the model. However, many of the state variables may not be of interest to the goal at hand. The computational work involved in the generation of values of all the state variables may be reduced if we have a *goal-directed* simulation strategy, and a representation which helps in identifying the dependencies and focusing the simulation.

3. *Flexibility regarding detail.* Human reasoning flexibly integrates computations of different degrees of accuracy and precision in a goal-directed way.

In the FR of a device, the CPD corresponding to a function of a device is a packaged simulation that is goal-directed and that bridges component- and device-level abstractions so as to understand the aspects of behavior relevant to the device function. The annotations of domain laws can provide pointers to numerical calculations. If the nodes of the graph are encoded with appropriate information about ranges of values and the annotations refer to component functions or domain laws in an appropriately parametrized way, the CPD can be viewed as a compact representation of a range of behaviors. Sticklen and his associates (see [17], [21] and [25] for some examples) and Toth [27] have built a number of complex systems that perform simulations of devices using FR. Keuneke [16] shows how FR of a device can be used to construct explanations of malfunction behavior by constructing a simulation of the device under various malfunction hypotheses.

3.2. Diagnosis

The first use to which FR was put was in diagnostic reasoning. In Sembugamoorthy and Chandrasekaran [19] a diagnostic knowledge structure was compiled for an electronic buzzer from its FR. The diagnostic knowledge structure was a malfunction tree, with a set of diagnostic rules for each of the malfunctions. Sticklen [20] used a similar idea for generating missing diagnostic knowledge. (References [2], [20], and [21] provide a sampling of papers that discuss various issues in using FR for diagnosis.)

The central idea in the application of FR for diagnosis can be summarized as follows. For simplicity, let us first consider a CPD in which each transition has only one annotation. Consider a transition in a CPD of the form

$$n_1 \xrightarrow{\texttt{By-function}\,F\texttt{-of-component}\ c} n_2$$

Suppose the device is in the partial state n_1, i.e., the device is in a state that satisfies the predicates corresponding to n_1. Suppose we test the device and observe that the device fails to reach n_2. What conclusions can we draw? Because the CPD asserts that the device goes from partial state n_1 to n_2 because of the function F of component c, we can hypothesize that the failure to reach n_2 is due to the component c not delivering the function F. Corresponding to this transition we can identify a possible malfunction state "Component C not delivering function F." The diagnostic rule "device satisfies n_1 but not n_2," can be used to establish this malfunction mode of the device. If the annotation had instead been "`By-CPD` *CPD-1*," where *CPD-1* is a specific CPD, we could similarly examine *CPD-1* to see why this transition failed (some transition in *CPD-1* should fail if the transition from n_1 to n_2 failed). Ultimately, we can identify some function of some component that would have to be responsible for the failure of the device to reach n_2.

The technique of identifying a component malfunction either directly from the annotation `By-Function` or by recursive application of `By-CPD` leads to a diagnostic tree that will have as tip nodes malfunctions of components or subcomponents. The diagnostic rule for each malfunction will be composed of rules of the form "If the predicates corresponding to node n_i are true, but those corresponding to n_j are not true, then.."

Not all diagnostic knowledge can be derived from design information alone, e.g., knowledge of likelihoods of failure — information extraneous to FR — can be used to order diagnostic hypotheses. Conversely, not all diagnostic knowledge derived from causal models is directly usable, since some variables mentioned in the diagnostic rules generated from causal models may not be directly observable. Additional inference may be required.

Diagnosis of computer programs. Allemang [1] has built a system that uses an FR of the underlying algorithm to debug computer programs intended to implement the algorithm. Allemang's work also illustrates another important point: that the logic of understanding how something works applies just as well to devices which are "abstract," like computer programs. Even though computer programs do not have physical components, they have components nevertheless and they play causal roles in achieving their functions just as much as physical components do in physical devices.

3. 3. Other uses

Design. FR is useful in a number of subtasks of design. Iwasaki, et al [13] and Vescovi, et al [28] describe a design framework in which again FR plays an important role, in particular for design verification. Chandrasekaran, Goel and Iwasaki [3] discuss the use of FR for representing causal aspects of a design rationale, i.e., explanations of the design choices. Goel and his students (see [9],

[10], [18], [23] and [24] for a sample of their work) have used FR to represent cases in a design library in the context of case-based design. This enables them to devise techniques that retrieve relevant designs as well as to critique a retrieved design for why it doesn't meet the current needs.

Device Libraries. The FR framework leads to the prospect of technical domain libraries of generic devices. Device classes at different levels of system description would be represented along with parametrized structural representation and the corresponding CPD's. Specific device representations can be constructed by choosing, instantiating and composing elements from the library. In cases where the design is novel, new CPD's can be composed. Josephson [14] reports on the use of abstract data types as the basis for building such device libraries. The Kritik system of Goel [9] uses a library of about 25 designs for case-based design.

Generating FRs for new devices. In all of our discussion so far, we have assumed that the designer or someone else has constructed an FR for the device of interest. Human experts, however, are often able to construct functional representations of devices that they have not seen before in the domains in which they have expertise. The same structural description can be associated with many different behaviors, only some of which may have been of interest to the designer. Thus, constructing an FR as intended by the designer will typically call for some form of abductive reasoning in which, using a number of cues, a hypothesis is made about the intended function of the device and a corresponding FR is constructed.

For the problem of constructing FR's for novel devices, Thadani [26] proposes a set of techniques. A central idea is that expertise in domains partly consists of *structure-function-CPD templates* at various levels of description. These templates consist of structural skeletons, functions that they can help achieve, and an abstract CPD that describes how the structural skeleton might achieve that function. Phrases such as "skeleton," "template" and "abstract" in the above are intended to refer to the fact that the templates and the CPD's may refer to classes of objects and behaviors, and also may not have all the details filled in. They are simply organized fragments of knowledge about the domain, fragments embodying pieces of understanding about structural configurations and their relations to various functions.

As a new physical situation is presented, the reasoning proceeds as follows. Templates from memory are matched to the description. All templates that match parts or the whole of the device are retrieved and ranked according to the degree of match. The templates that have the highest degree of match are considered first. The templates are instantiated with as much detailed information from the device as available. Instantiated CPD's may suggest additional hypotheses about the possible roles of other structural parts. These hypotheses may be partially or completely verified by checking the conditions associated with the selected CPD's. The hypotheses may also be verified by simulating the CPD with instantiated parameters. In this process, additional hypotheses might be generated. If the predictions are confirmed, that part of the device is labeled with the function from the template. When the cycle of the identifications and verifications is over, we may have a set of alternate hypotheses for parts of device. Each consistent set of interpretations gives rise to labeling the parts of the structure differently. The relabeling for a specific interpretation changes the structural description, raising the level of abstraction in which the structure is described.

This relabeling enables a new round of matchings to be activated, and a new set of structure-function templates to be retrieved. Each stage prunes away some of the hypotheses from the previous levels, and might add a few hypotheses, but in general, with a sufficient body of domain knowledge in the form of templates and their constraints, the number of possible interpretations converges to a small number of highly plausible and consistent ones.

The above picture of top-down recognition alternating with bottom-up hypothesis verification is one which I think models our general reasoning about the behavior of the physical world. In this view, we are armed with a large library of skeletal FR's which relate behaviors, structural constraints and CPD's at various levels of abstraction in a given domain. Use of a large repertoire of FR's, spanning multiple levels of abstraction and goals, gives the agent the capability for very goal-directed and efficient simulation of just the relevant parts of the world for predicting behavior.

4. Concluding Remarks

The real power of intelligent behavior arises from the agent's ability to organize reasoning and computational resources in a goal-directed way, and qualitativeness of reasoning is just one aspect of it. FR investigates issues about how causal knowledge is indexed and packaged functionally. This work is complementary to the work on qualitative device models. Together they provide an integration of top-down and bottom-up reasoning techniques for efficient goal-directed causal reasoning. The research area is rich in topics for further expansion and exploitation. The FR framework is just part of a larger framework, one in which reasoning, action and perception are seen as an integrated whole.

Acknowledgments

I acknowledge the support of ARPA in the preparation of this paper (Arpa order no. A714, USAF Contract no. F30602-93-c-0243, monitored by Rome Laboratories) I also acknowledge the collaboration over a number of years of many colleagues, both at Ohio State University and elsewhere, in developing these ideas.

References

1. Allemang, D.: Using functional models in automatic debugging. IEEE Expert. **6** (1991), 13-18
2. Chandrasekaran, B., Smith, J. W. Jr., Sticklen, J.: 'Deep' Models and their relation to diagnosis. Artificial Intelligence in Medicine **1** (1989) 29 -40
3. Chandrasekaran, B., Goel, A., Iwasaki, Y.: Functional representation as design rationale. IEEE Computer, Special Issue on Concurrent Engineering (1993) 48-56
4. Chandrasekaran, B.: Functional representation and causal processes. In Advances in Computers **38** (1994) 73-143. Academic Press
5. Chittaro, L., Tasso, C., and Toppano, E., Putting Functional Knowledge on Firmer Ground. In *Reasoning About Function*, Amruth N. Kumar, ed., American Association for Artificial Intelligence Workshop Program (1993) 23-30
6. de Kleer, J. Brown, J.S.: A qualitative physics based on confluences. Artificial Intelligence, **24** (1984) 7-83
7. Forbus, K. : Qualitative process theory. Artificial Intelligence **24** (1984) 85-168
8. Forbus, K..: Qualitative physics: Past, present and future. Exploring Artificial Intelligence, H. Shrobe, ed., San Mateo, CA: Morgan Kauffman (1988) 239-296
9. Goel, A.K..: Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph. D. thesis, The Ohio State University, LAIR (1989)
10. Goel, A.K.. : A model-based approach to case adaptation. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, August 7-10 (1991) 143-148, Hillsdale, NJ: Lawrence Erlbaum
11. Hodges, J.: Naive mechanics: a computational model of device use and function in design improvisation. IEEE Expert **7** (1992) 14-27
12. Iwasaki, Y., Chandrasekaran, B.: Design verification through function- and behavior-oriented representations: Bridging the gap between function and behavior. *Artificial Intelligence in Design '92*, John S. Gero, ed., Kluwer Academic Publishers (1992) 597-616
13. Iwasaki, Y., Fikes, R., Vescovi, M., Chandrasekaran, B.: How things are intended to work: Capturing functional knowledge in device design. In *Proceedings of the 13th International Join Conference of Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann (1993) 1516-1522
14. Josephson, J. R.: The Functional Representation Language FR as a Family of Data Types, The Ohio State University, Laboratory for Artificial Intelligence Research, Columbus, OH, tech report (1993)
15. Keuneke, A., Allemang, D.: Understanding devices: Representing dynamic states. Technical Report, The Ohio State University, Laboratory for Artificial Intelligence Research, Columbus, OH (1988)

16. Keuneke, A.: Machine Understanding of Devices: Causal Explanation of Diagnostic Conclusions. Ph. D thesis. The Ohio State University (1989)

17. Pegah, M., Sticklen, J., Bond, W.: Functional representation and reasoning about the F/A-18 aircraft fuel system. IEEE Expert 8 (1993) 65-71

18. Prabhakar, S. and Goel, A. K.: Integrating case-based and model-based reasoning for creative design: constraint discovery, model revision and case composition. In *Proceedings of the Second International Conference on Computational Models of Creative Design*, Heron Island, Australia, (1992) Kluwer Academic Press

19. Sembugamoorthy, V., and Chandrasekaran, B.: Functional representation of devices and compilation of diagnostic problem-solving systems. In *Experience, Memory, and Learning*, J. Kolodner and C. Reisbeck, editors, Lawrence Erlbaum Associates (1986) 47-73

20. Sticklen, J.H.: MDX2, an integrated medical diagnostic system, Ph. D. dissertation, The Ohio State University, Columbus, OH (1987)

21. Sticklen, J., Kamel, A., & Bond, W. E. (1991): Integrating quantitative and qualitative computations in a functional framework. Engineering Applications of Artificial Intelligence 4(1) (1991) 1-10.

22. Sticklen, J., & Tufankji, R.: Utilizing a functional approach for modeling biological systems. Mathematical and Computer Modeling 16 (1992) 145-160

23. Stroulia, E., Shankar, M., Goel, A.K., and Penberthy, L..: A model-based approach to blame assignment in design. In *Proceedings of the Second International Conference on AI in Design.*, Kluwer Academic Press (1992) 519-538

24. Stroulia, E., and Goel, A.K.: Generic teleological mechanisms and their use in case adaptation. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (1992) 319-324

25. Sun, J., and Sticklen, J.: Steps toward tractable envisionment via a functional approach. In *The Second AAAI Workshop on Model Based Reasoning*, American Association for Artificial Intelligence, Boston (1990) 50-55

26. Thadani, S.: Constructing Functional Models of a Device from Its Structural Description. Ph. D Thesis, Department of Computer & Information Science, The Ohio State University (1994)

27. Toth, S.: Using Functional Representation for Smart Simulation of Devices, Ph. D Thesis, Department of Computer & information Science, The Ohio State University (1993)

28. Vescovi, M., Iwasaki, Y., Fikes, R., and Chandrasekaran, B.: CFRL: A language for specifying the causal functionality of engineered devices. In *Proceedings of the Eleventh National Conference on AI,* American Association for Artificial Intelligence (1993) 626-633. AAAI Press/MIT Press

How to Make Intelligent Digital Libraries

Edward A. Fox

Virginia Tech, Blacksburg VA 24061, USA

Abstract. Building digital libraries is one of the most important Grand Challenge problems faced by information professionals. This paper surveys the relevant literature and includes recommendations for research, based on our experience with a digital library for computer science and its use in improving undergraduate education. Of particular interest is the application of intelligent system methods to help provide an effective solution. Mediators or agents are needed throughout the system. It is also important to improve the interface, help with searching, and in general allow the digital library to carry out many of the functions of an expert human intermediary.

1 Introduction

In 1965, J.C.R. Licklider described a future distributed electronic library that would cover the globe, providing widespread access and assistance to humans interesting in useful information sources [45]. Today, we are implementing his vision on an unprecedented scale, using nearly ubiquitous computing and communication capabilities that eventually will be able to support and extend mankind's needs for library information. This Grand Challenge to the information technology community will lead to a variety of activities over the next several decades. Here we focus on those needed to develop *intelligent* digital libraries.

One can arrive at the idea of digital libraries from many different directions. Thus, Nelson focused on an open world of connected information systems, with links providing the basis for associations and navigation, as in *hypertext* and *hypermedia* bases [52]. At Brown University this led to systems and collections of information oriented to aid education and research, and eventually to the important concept of a *link engine* to be supported by operating systems [34]. Expanding this to the global scale, as originally envisioned by Nelson, the World-Wide Web [7] has spread to millions of computers. Tools like NCSA Mosaic, and more sophisticated client and server implementations like Hyper-G [41], have given a rapidly growing user base a first generation, in relatively usable form, of a partial global digital library. These have emerged hand-in-hand with search systems and the network of Wide Area Information Servers and software [38].

Simultaneously, the electronic publishing world has articulated visions of a new, flexible and rapid form of information production and dissemination that has caught the attention of disciplines as diverse as psychology [28] and computer science [14, 13]. While professional associations were among the first to explore the possibilities, commercial publishers also have embarked upon large-scale experiments (e.g., the TULIP project of Elsevier, and the Red Sage effort that involves Springer-Verlag and AT&T Bell Laboratories).

For current pilot studies to be successful, however, solutions are needed for problems related to copyright, royalty management, and other intellectual property rights matters. This point is so central that it was and still remains one of the key concerns of the Corporation for National Research Initiatives. Early work in this area led to coining of the term *knowbot* and the phrase *digital library* [40]. Clearly, changes are needed regarding management of copyright [29]. However, rather than people paying for each access or use, it is likely that subscription-based approaches [57] will be more suitable, at least in certain contexts like education.

One also arrives at the idea of digital libraries from working with multimedia technology [15]. Here the quantities of data are boundless, storage requirements (even with compression) are enormous, and the network bandwidth for multiple simultaneous access by large populations seems unachievable. Clearly, hierarchical or similar storage, caching, and distribution schemes like those suggested by Licklider seem to force us toward digital libraries in this domain! And, the relatively rapid emergence and acceptance of standards (e.g., for compression of still images and video, using JPEG and MPEG) [16], facilitates the kind of interoperability one associates with libraries.

From these many directions we move toward Networked Multimedia Information Access [17] and the emerging infrastructure of Digital Libraries. Disciplines that play a key part include: computational linguistics, electronic publishing, human-computer interaction, hypertext, hypermedia, information retrieval, information science, library science, multimedia and networking.

2 Grand Challenge

Building the Global Digital Library is one of the most important Grand Challenges we face. Doing so will solve many troublesome current problems.

One such problem is the fundamental human need to obtain useful information as required. A model of the process involved is desired, and has been partially specified in works such as [6] where a functional decomposition is proposed. That, in turn, suggests a *distributed expert-based information system* design which has been partially specified and implemented [4]. This enterprise is especially difficult; humans often have Anomolous States of Knowledge, which we sometimes interpret as needs or problems, but these are hard to characterize, depending in part on complex situations of uncertainty, ambiguity, partial knowledge, and missing information. All of the Artificial Intelligence problems of knowledge representation, natural language understanding, planning, learning, etc. are in evidence here!

This problem can be redefined in terms of computers carrying out the activities of expert librarians. Many studies have considered applying expert system techniques to library situations [51]. Some success has been achieved in helping automate the referral process [69]. More to the point, but very hard, is the problem of modeling human intermediaries [8].

A second problem is that existing information systems have been built in isolation, using incompatible methods. Barriers due to proprietary solutions, model mismatches, or even deliberately imposed firewalls have led to extensive work on federated or heterogeneous databases, and eventually to the elegant approach of mediators [71]. Yet, this concept has still to be extensively applied. Further, the technology of object-oriented DBMS certainly relates, but in some senses complicates the matter further, due to lack of standard query languages and methods. We see the importance of OODBMS for managing the text and other objects needed for storing and analyzing digital library contents [27], but have yet to achieve interoperability or even easy interchange.

A third problem is that of scaling up services to handle larger quantities of information, larger information objects, and larger numbers of users. Even if we have small objects, as in hypertext, the problem is serious, demanding careful attention to system performance, network topology, and server capabilities [1]. Many digital library efforts work with bitmap page images [35] and those demand large amounts of storage (e.g., in the TULIP project, it is estimated that an average journal would consume roughly 1/4 gigabyte each year)! Further, extensive and expensive processing is needed if we are to undertake a retrospective conversion from paper to electronic forms [66]. With multimedia information, such as digital video that can often be very helpful for education, special attention and *smart* handling is needed to provide adequate quality of service [19].

A fourth problem is that of revitalizing education — that may be eased with digital libraries. Providing interactive access to primary materials, with instructors outfitted with suitable *authoring* tools, should allow easy adding of value, re-thinking, reorganizing, and enhancing courses and curricula [24]. This extends to the level of campuses and universities, that are too short on space and funds to extend conventional libraries, and so reshape the role of interns, textbooks, and information technology organization to suit [56]. Closely related to this is *information overload* where a person needs help managing the information received (and sought out). We need tools able to learn about a user to help with this important problem [48]. (It is growing more severe daily as email services proliferate!)

Finally, the information overload problem has a generalization in the problem of developing suitable interfaces to digital libraries. How can one have search interfaces that run on a variety of platforms [39]? How can powerful multimedia systems be used with large amounts of incoming information? How can workstations coordinate the presentation of multimedia information to adapt to users' preferences, capabilities, challenges, styles, and tasks [2]?

Seeing this great challenge, NSF began investigating whether there was need for special funding in the digital library area [18]. That led to an NSF/ARPA/NASA call for research in this area [55]. In addition, there are various other U.S. initiatives and legislation [25], as well as efforts in many other nations, including France, Japan and Singapore.

3 Background Literature

A great deal of research has been undertaken relating to digital libraries. We focus here on several problems regarding intelligent information systems.

One key area with many such problems is that of intelligent information retrieval. This builds upon decades of investigation of information retrieval [62, 63]. For example, one problem is to characterize (model) a user to allow adaptation, and to learn about user interests and shifts so as to modify system response. Another problem: How can the meaning of a query and of each document or even passage in a large collection be captured and compared? What intelligent processing at the level of morphology, lexicons, phrases, semantic relations, etc. can be carried out on very large collections and be shown to improve retrieval? For example, can short user queries be disambiguated correctly, and possibly expanded with closely related terms, to facilitate matching? A large study of this type was undertaken as part of the TIPSTER program of ARPA, and the investigators hope for additional support to carry on their efforts [46].

A second key area is modeling the information retrieval process. At the highest level is the matter of how users, human intermediaries, and systems inter-relate. An important method is to study interactions between users and intermediaries, and to characterize and model those [36]. One of the central problems in that regard is how to model and deal with uncertainty, ambiguity, and parallel representations [37]. Modeling the information retrieval domain is particularly complex because we do not know why different search approaches each yield fairly good but largely non-overlapping retrieval results. On the operational side, it turns out that combining results from various approaches (or searchers or search methods or subcollections) can be done in several ways that lead to better results than any of the single runs [3, 5].

A promising generalization of the modeling problem is to apply logics [60, 61]. Thus, one tries to determine relevance of an information item given a user's knowledge state, using probabilities, beliefs or other measures, considering context, situation, and possible worlds. One recent suggestion is to use probabilistic terminological logic [65]. Another suggestion is to use belief revision and agents to develop a complete model, not only of the retrieval system but also of the interaction with the user [47]. Good results have been achieved already using probabilistic inference networks [68].

Another problem, more focused, is to develop methods to categorize or classify items (e.g., routing messages or mail to the right parties). Here, learning techniques are crucial. Good results have been achieved with probabilistic classifiers [44] as well as with an expert network [72].

Numerous other studies have been undertaken. The sensible methodology for such complex enterprises as building digital libraries, then, is to determine general requirements, prepare a taxonomy, develop an appropriate architecture, find specific requirements for each component in the architecture, and then implement each component in a way that allows easy change or replacement [32, 31].

4 Prior Work

In keeping with this recommended methodology for building digital libraries, our own research has moved in that direction. Our early efforts were focused on implementing a system that could handle various types of documents (including especially those with structure and components), and which carried out the functions of an intermediary [11, 12]. It became clear, however, that in addition to knowledge representation support for frames [70], it was also important to have persistent storage of large semantic networks (as found in lexicons, and hyperbases) [20]. To focus this work, and reach production quality, we implemented the MARIAN system for online public access to a library catalog [21], supporting the search functions for a digital library.

To develop a prototype digital library in computer science, then, we worked on data collection/transformation/representation, interface development, and search support [22]. The currently running *Envision* system has extensive information from ACM, uses the MARIAN search engine, and has a powerful interface for query entry and result management and visualization [54]. In keeping with the approach to digital library construction mentioned above, we elected to integrate the Envision software with readily available networked hypermedia software in the World-Wide Web. In particular, we now use NCSA Mosaic, but are investigating use of Hyper-G [41] as well.

Another key area of our work that relates to intelligent information retrieval is that of combining results from multiple search methods and systems. After several years of exploration with the very large *TREC* collection of millions of records, we have had good results [23, 26, 5]. In particular, the fused or combined results are generally better, almost always significantly better according to statistical tests, than the average results of any of the individual methods.

5 Recommendations

Based on our prior work, and study of other efforts that relate to digital libraries, we offer a series of recommendations.

First, we encourage the use of mediators or agents, in a distributed information environment. There are many issues related to the use of agents [58]. Of particular concern are [64]: metaphors, user control, inspectability, adaptivity, security, resiliency, inter-agent communication, handling of existing active features and extensibility. It is crucial that agents fit into the overall AI and human interaction schemes we desire [49] and behave in ways to satisfy users and their concerns [53]. Agents should be connected into the interface and task development processes [42] and must learn about the users and how they carry out their tasks [48, 50].

The behavior of agents and their interaction is governed in part by the knowledge available. In the role of mediators, they are keyed to the type of data, information, or knowledge they manage [67]. Large shared knowledge bases can facilitate cooperative activity among a set of agents [33]. Agents that allow various knowledge representations are better able to undertake complex activities since each of those activities can be modeled with the most appropriate representation scheme [59]. Communication among agents is facilitated, and human development of interoperable agents is enhanced, through the use of a common knowledge representation language with clear semantics (e.g., using KQML and KIF) [30]. However, information will at times be incomplete, or uncertain, and so agents for such complex activities as internet or digital library access had best have flexible planning capabilities [9].

Second, we point to outside publications that have insightful comments, and urge that these be considered carefully. Thus, in keeping with our Envision project theme, we encourage user-centered construction of digital libraries, with careful design for usability, considering the broad human and social infrastructure surrounding the system [43]. Also, in keeping with our work on LEND, we encourage use of an OODBMS, suitably tailored [27]. Further, given the large hyperbase that emerges in connection with digital libraries, we encourage automatic generation of as many high-value links as possible [10].

Third, we give some general recommendations regarding the use of intelligent information system techniques for digital libraries.

1. Searching is harder for users than browsing or following hypertext links, and so between these, information retrieval should be the main focus of application for intelligent methods.
2. Intelligent methods are important in the interface, as long as users have control, and can get explanations as needed. Of particular importance is hiding complexity, setting numerous parameters/defaults, and adapting to capabilities when that leads to significantly more efficient or easier interaction.
3. Don't do stupid things — at any level of processing. Thus, look for failures, and learn about users' difficulties, correcting the situation as soon as possible. This covers cases such as demanding long sequences of keystrokes when the full sequence is always done if the first is selected, or reducing the name *AT&T* into the two components *at* and *t*, both of which are likely to be on a retrieval stop word list.
4. Remember that users like to organize as they read and learn, and must have tools inside the system for that. Also, digital libraries must be dynamic, since many people have a strong drive to self-publish. Therefore, provide advice and guidance to users in these processes, since good style and structuring ability are rare qualities in users.
5. Users need mental models about system behavior, so provide and reinforce these, at the global (architecture) level as well as at the level of components (e.g., agents). Develop and refine these through formative evaluation.
6. Work with others to build integrated digital libraries, where your expertise leads to those components or agents you are best qualified to prepare. Make use of *standards* including ones like KIF and KQML [30].

Note that in [22] we gave general principles for constructing digital libraries, which we believe are still important, but do not repeat them here, in the interest of brevity. Also, in [18] there are many discussions and recommendations regarding research topics and policies about digital libraries.

6 Planned Work, Conclusions

Work on the Envision system is nearing fruition, and in Fall 1994 we will deploy it at Virginia Tech to help with our education project on *Interactive Learning with a Digital Library in Computer Science.* Deployment at Norfolk State University, and eventually through a server at ACM Headquarters, will expand the base of users.

At the same time we are expanding the content to include more publications, and more types and forms. Priorities in conversion are based on requirements or wishes for particular courses in which we first deploy our system and educational approaches.

In addition to making use of Mosaic and/or Hyper-G [41], we will use the KMS (hypertext and computer-supported cooperative work) system in our classes. Various small *agents* in that environment are being collected. We also expect to collect agents and tools developed elsewhere.

Already, instructors have requested a number of tools to help with courseware preparation and use of the digital library. We have begun developing additional conversion tools, and in several cases have identified situations where *intelligent* processing is called for. It seems clear that use of intelligent system methods will be crucial in digital libraries, but that we are just at the beginning of the process of identifying requirements and capabilities, and trying to add them to the overall architecture.

Acknowledgments. The digital library research at Virginia Tech has been funded in part by the National Science Foundation through a series of grants, most recently for *A User Centered Database from the Computer Science Literature, Interactive Learning with a Digital Library in Computer Science* and *Interactive Accessibility: Breaking Barriers to the Power of Computing.* Thanks go to all of the co-investigators on these projects! PRC Inc. and ARPA have funded other efforts, especially our activities for TREC-1 and TREC-2 [23, 26], and the co-authors have provided special assistance. The Virginia Tech Computing Center and College of Arts and Sciences have provided cost sharing, and in the former case, staffing to help with the MARIAN and Envision projects. Our Digital Library work has involved many faculty and students, especially those who have co-authored key works [21, 22, 24, 19, 54]. Our thanks go to all who have played a role. Keith Tomlin provided extensive editorial assistance in preparing [18]. Furthermore, we are grateful for the materials and assistance provided or offered by numerous publishers, most notably ACM, IEEE-CS, Springer-Verlag, Elsevier, and Addison-Wesley.

References

1. Robert M. Akscyn and Donald L. McCracken. PLEXUS: A hypermedia architecture for large-scale digital libraries. In *Proceedings of SIGDOC '93, the 11th Annual International Conference on Systems Documentation*, Waterloo, Ontario, October 5-8, 1993. 15 pages.

2. Yigal Arens and Eduard Hovy. Design considerations for model-based multimedia interaction managers. In Zahid Ahmed, Robert Akscyn, Nicholas Belkin, Edward Fox, and Scott Stevens, editors, *Position Papers, IEEE CAIA'94 Workshop on Intelligent Access to On-Line Digital Libraries*, San Antonio, TX, March 1, 1994. 12 pages.
3. Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings Seventeenth Annual International ACM-SIGIR Conference on R&D in Information Retrieval, SIGIR '94*, pages 173–181, Dublin, Ireland, July 1994. Springer-Verlag.
4. N. J. Belkin, C. Borgman, H. Brooks, T. Bylander, W. B. Croft, P. Daniels, S. Deerwester, E. A. Fox, P. Ingwersen, R. Rada, K. Sparck Jones, R. Thompson, and D. Walker. Distributed expert-based information systems: An interdisciplinary approach. *Information Processing & Management*, 23(5):395–409, 1987.
5. N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management*, 30:in press, 1994.
6. Nicholas J. Belkin, T. Seeger, and G. Wersig. Distributed expert problem treatment as a model for information system analysis and design. *Journal of Information Science*, 5:153–167, 1983.
7. T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollermann. WorldWideWeb: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2):52–58, Spring 1992.
8. H. Drenth, A. Morris, and G. Tseng. Expert systems as information intermediaries. *Annual Review of Information Science and Technology*, 26:133–154, 1991.
9. Oren Etzioni and Daniel Weld. A softbot-based interface to the internet. *Communications of the Association for Computing Machinery*, 37(7):72–76, July 1994.
10. Will Fitzgerald and Christopher Wisdo. Using natural language processing to construct large-scale hypertext systems. In Zahid Ahmed, Robert Akscyn, Nicholas Belkin, Edward Fox, and Scott Stevens, editors, *Position Papers, IEEE CAIA'94 Workshop on Intelligent Access to On-Line Digital Libraries*, San Antonio, TX, March 1, 1994. 18 pages.
11. E. A. Fox and Robert K. France. Architecture of an expert system for composite document analysis, representation and retrieval. *International Journal of Approximate Reasoning*, 1(1):151–175, 1987.
12. Edward A. Fox. Development of the CODER system: A testbed for artificial intelligence methods in information retrieval. *Information Processing & Management*, 23(4):341–366, 1987.
13. Edward A. Fox. ACM Press Database and Electronic Products – new services for the information age. *Communications of the Association for Computing Machinery*, 31(8):948–951, 1988.
14. Edward A. Fox. How to proceed toward electronic archives and publishing. *Psychological Science*, 1(6):355–358, November 1990.
15. Edward A. Fox. Advances in interactive digital multimedia systems. *IEEE Computer*, 24(10):9–21, October 1991.
16. Edward A. Fox. Guest editor's introduction: Standards and the emergence of digital multimedia systems. *Communications of the Association for Computing Machinery*, 34(4):26–29, April 1991. For Special Section on Digital Multimedia Systems.

17. Edward A. Fox. From information retrieval to networked multimedia information access. In G. Knorz, J. Krause, and C. Womser-Hacker, editors, *Proc. der 1. Tagung Information Retrieval '93*, pages 116–124, University of Regensburg, Germany, Univ. of Konstanz Press, 13-15 September 1993.

18. Edward A. Fox. Source book on digital libraries. Technical Report TR 93-35, Virginia Tech, Department of Computer Science, Blacksburg, VA, December 1993. Edited volume, available by anonymous FTP from directory pub/DigitalLibrary on fox.cs.vt.edu.

19. Edward A. Fox and Ghaleb Abdulla. Digital video delivery for a digital library in computer science. In *Proc. High-Speed Networking and Multimedia Computing Workshop, IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, CA, Feb. 6-10 1994. 7 pages.

20. Edward A. Fox, Qi Fan Chen, and Robert K. France. Integrating search and retrieval with hypertext. In Emily Berk and Joseph Devlin, editors, *Hypertext/Hypermedia Handbook*, pages 329–355. McGraw-Hill, Inc., New York, 1991.

21. Edward A. Fox, Robert K. France, Eskinder Sahle, Amjad Daoud, and Ben E. Cline. Development of a modern OPAC: From REVTOLC to MARIAN. In *Proc. SIGIR 93, 16th Annual Int'l ACM SIGIR Conference on R&D in Information Retrieval*, pages 248–259, Pittsburgh, PA, June 27-July 1, 1993.

22. Edward A. Fox, Deborah Hix, Lucy T. Nowell, Dennis J. Brueni, William C. Wake, Lenwood S. Heath, and Durgesh Rao. Users, user interfaces, and objects: Envision, a digital library. *Journal of the American Society for Information Science*, 44(8):480–491, September 1993.

23. Edward A. Fox, M. Prabhakar Koushik, Joseph Shaw, Russell Modlin, and Durgesh Rao. Combining evidence from multiple searches. In Donna K. Harman, editor, *The First Text REtrieval Conference (TREC-1)*, pages 319–328, Gaithersburg, MD, March 1993. NIST Special Publication 500-207.

24. Edward A. Fox, J.A.N. Lee, Clifford Shaffer, H. Rex Hartson, and Dwight Barnette. Interactive learning with a digital library in computer science, March 1993. (Funded) Education Infrastructure Proposal to the NSF for 1993-96.

25. Edward A. Fox and Lois F. Lunin. Introduction and overview to perspectives on digital libraries. *Journal of the American Society for Information Science*, 44(8):441–445, September 1993.

26. Edward A. Fox and Joseph Shaw. Combination of multiple searches. In Donna K. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 243–252, Gaithersburg, MD, March 1994. NIST Special Publication 500-215.

27. Robert P. Futrelle and Xiaolan Zhang. Large-scale persistent object systems for corpus linguistics and information retrieval. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 80–87, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atgl.WUSTL.edu/DL94.

28. William Gardner. The electronic archive: Scientific publishing for the 1990s. *Psychological Science*, 1(6):333–341, November 1990.

29. John R. Garrett and Patrice A. Lyons. Toward an electronic copyright management system. *Journal of the American Society for Information Science*, 44(8):468–473, September 1993.

30. Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the Association for Computing Machinery*, 37(7):48–53, July 1994.

31. Henry M. Gladney, Zahid Ahmed, Ron Ashany, Nicholas J. Belkin, Edward A. Fox, and Maria Zemankova. Digital library: Gross structure and requirements (report from a March 1994 Workshop). Technical Report IBM Research Report RJ9840 and Virginia Tech Dept. of Computer Science TR 94-25, IBM Almaden Research Laboratory, June 1994. Available by anonymous FTP from directory pub/DigitalLibrary on fox.cs.vt.edu.
32. Henry M. Gladney, Edward A. Fox, Zahid Ahmed, Ron Ashany, Nicholas J. Belkin, and Maria Zemankova. Digital library: Gross structure and requirements: Report from a March 1994 Workshop. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 101–107, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atg1.WUSTL.edu/DL94.
33. R. V. Guha and Douglas B. Lenat. Enabling agents to work together. *Communications of the Association for Computing Machinery*, 37(7):126–142, July 1994.
34. Bernard J. Haan, Paul Kahn, Victor A. Riley, James H. Coombs, and Norman K. Meyrowitz. IRIS hypermedia services. *Communications of the Association for Computing Machinery*, 35(1):36–51, January 1992.
35. Melia M. Hoffman, Lawrence O'Gorman, Guy A. Story, James Q. Arnold, and Nina H. Macdonald. The RightPages service: An image-based electronic library. *Journal of the American Society for Information Science*, 44(8):446–452, September 1993.
36. Peter Ingwersen. *Intermediary Functions in Information Retrieval Interaction*. PhD thesis, Copenhagen Business School, Institute of Informatics and Management Accounting, 1991.
37. Peter Ingwersen. Polyrepresentation of information needs and semantic entities: Elements of a cognitive theory for information retrieval interaction. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings Seventeenth Annual International ACM-SIGIR Conference on R&D in Information Retrieval, SIGIR '94*, pages 101–110, Dublin, Ireland, July 1994. Springer-Verlag.
38. Brewster Kahle. An information system for corporate users: Wide area information servers. *ONLINE Magazine*, August 1991. Also as Thinking Machine technical report TMC-199.
39. Brewster Kahle, Harry Morris, Jonathan Goldman, Thomas Erickson, and John Curran. Interfaces for distributed systems of information servers. *Journal of the American Society for Information Science*, 44(8):453–467, September 1993.
40. Robert Kahn and Vint Cerf. The Digital Library Project. Volume 1: The world of knowbots. Technical report, Corporation for National Research Initiatives, Reston, VA, 1988.
41. F. Kappe, H. Maurer, and N. Sherbakov. Hyper-G – a universal hypermedia system. *Journal of Educational Multimedia and Hypermedia*, 2(1):39–66, 1993.
42. Henry A. Kautz, Bart Selman, and Michael Coen. Bottom-up design of software agents. *Communications of the Association for Computing Machinery*, 37(7):143–147, July 1994.
43. Rob Kling and Margaret Elliott. Digital library design for usability. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 146–155, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atg1.WUSTL.edu/DL94.

44. David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings Seventeenth Annual International ACM-SIGIR Conference on R&D in Information Retrieval, SIGIR '94*, pages 3–12, Dublin, Ireland, July 1994. Springer-Verlag.

45. J. C. R. Licklider. *Libraries of the Future.* The MIT Press, Cambridge, MA, 1965.

46. Elizabeth D. Liddy, Michael B. Eisenberg, Charles R. McClure, Kim Mills, Susan Mernit, and James D. Luckett. Research agenda for the intelligent digital library. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 12–20, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atg1.WUSTL.edu/DL94.

47. Brian Logan, Steven Reece, and Karen Sparck Jones. Modelling information retrieval agents with belief revision. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings Seventeenth Annual International ACM-SIGIR Conference on R&D in Information Retrieval, SIGIR '94*, pages 91–100, Dublin, Ireland, July 1994. Springer-Verlag.

48. Pattie Maes. Agents that reduce work and information overload. *Communications of the Association for Computing Machinery*, 37(7):30–40, July 1994.

49. Marvin Minsky and Doug Riecken. A conversation with Marvin Minsky about agents. *Communications of the Association for Computing Machinery*, 37(7):22–29, July 1994.

50. Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the Association for Computing Machinery*, 37(7):80–91, July 1994.

51. A. Morris. Expert systems for library and information services - a review. *Information Processing & Management*, 27(3):713–724, 1991.

52. T.H. Nelson. *Literary Machines.* The Distributors, South Bend, IN, 1981.

53. Donald A. Norman. How might people interact with agents. *Communications of the Association for Computing Machinery*, 37(7):68–71, July 1994.

54. Lucy Nowell, Edward A. Fox, Lenwood Heath, Deborah Hix, William Wake, and Eric Labow. Seeing things your way: Information visualization for a user-centered database of computer science literature. Technical Report TR 94-06, Department of Computer Science, Virginia Polytechnic Institute and State University, 1994.

55. NSF. Research on digital libraries. Technical Report NSF 93-141, National Science Foundation, 1993.

56. Larry Press. Tomorrow's campus. *Communications of the Association for Computing Machinery*, 37(7):13–17, July 1994.

57. Gregory J. E. Rawlins. Publishing over the next decade. *Journal of the American Society for Information Science*, 44(8):474–479, September 1993.

58. Doug Riecken. Introduction to special issue on intelligent agents. *Communications of the Association for Computing Machinery*, 37(7):18–21, July 1994.

59. Doug Riecken. M: An architecture of integrated agents. *Communications of the Association for Computing Machinery*, 37(7):106–116, July 1994.

60. C. J. Van Rijsbergen. A new theoretical framework for information retrieval. In *Proc 1986 ACM Conf. on Research and Development in Information Retrieval*, pages 194–200, Pisa, Italy, September 1986.

61. C. J. Van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29:481–485, 1986.

62. C.J. Van Rijsbergen. *Information Retrieval: Second Edition.* Butterworths, London, England, 1979.

63. Gerard Salton. *Automatic Text Processing.* Addison-Wesley, 1989.

64. J. Alfredo Sánchez. User agents in the interface to digital libraries. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 217–218, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atgl.WUSTL.edu/DL94.

65. Fabrizio Sebastiani. A probabilistic terminological logic for modelling information retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings Seventeenth Annual International ACM-SIGIR Conference on R&D in Information Retrieval, SIGIR '94*, pages 122–130, Dublin, Ireland, July 1994. Springer-Verlag.

66. Sargur N. Srihari, Stephen W. Lam, Jonathan J. Hull, Rohini K. Srihari, and Venugopal Govindaraju. Intelligent data retrieval from raster images of documents. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 34–40, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atgl.WUSTL.edu/DL94.

67. Richard M. Tong and David H. Holtzman. Knowledge-based access to heterogeneous information sources. In John L. Schnase, John J. Leggett, Richard K. Furuta, and Ted Metcalfe, editors, *Proceedings of Digital Libraries '94: The First Annual Conf. on the Theory and Practice of Digital Libraries*, pages 61–66, College Station, TX, June 19-21, 1994. Hypermedia Research Laboratory, Dept. of Computer Science, Texas A&M Univ. Electronic proceedings at http://atgl.WUSTL.edu/DL94.

68. Howard Turtle and W. Bruce Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

69. A. Vickery, H. M. Brooks, B. A. Robinson, and J. Stephens. Expert system for referral. Technical Report Library and Information Research Report 66, The British Library, 1988.

70. M. Weaver, R. France, Q. Chen, and E. Fox. Using a frame-based language for information retrieval. *International Journal of Intelligent Systems*, 4(3):223–257, 1989.

71. Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992. ISSN 0018-9162.

72. Yiming Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings Seventeenth Annual International ACM-SIGIR Conference on R&D in Information Retrieval, SIGIR '94*, pages 13–22, Dublin, Ireland, July 1994. Springer-Verlag.

"Some Methodology and Representation Problems for the Semantics of Prosaic Application Domains"

(*Extended Abstract*)

R.A. Meersman

INFOLAB, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands
email: Robert.Meersman@kub.nl
fax: (+31)13 663069

Introduction.

We shall not offer a formal definition for what constitute *prosaic* application domains. They are best described by a few examples: accounting domains, employee files, inventory control, bill-of-material applications, customer files, and the like. They usually have the property of having relatively simple data schemas, and large database populations; existing systems for such domains tend to be fairly old, because they solve mundane problems, that could be treated with early information systems technology, and often are of strategic importance, because the data are fundamental to the operation of a business or an organisation. The application programs surrounding such databases tend to be fairly *shallow,* i.e. their processing of the stored data is often minimal and does not depend on a high level of integration among these application programs. (This is not the same as saying that such integration is not required).

Also, the application program environments tend to be very large and organisationally very complex, as they have evolved over long periods of time. They form the favorite subjects of so-called *legacy systems* (Brodie [4]).

Another distinguishing aspect of prosaic domains is that the end-users of the corresponding systems usually are not very specialized experts; a fact usually reflected in a low satisfaction and high frustation rate because often it were the system developers who have tended to make implementation decisions with conceptual implication, in their place. It is precisely this last fact that makes prosaic domains into somewhat of an oxymoron, namely a relatively interesting topic for research. Indeed, their study raises fundamental semantic issues about the nature of their stored information, especially in a technological evolution where more and more of these systems need to be interconnected ('interoperation").

In what follows we shall discuss precisely which semantic issues are involved, as well as some of the implications for methodologies that must evolve such non expert users and must lead to large organizationally complex application environments that are essentially data driven. In particular, as with almost all legacy systems, the economic issues of re-use of existing components are a primary consideration that is closely linked to a proper semantical treatment of the information involved.

Semantic Issues in Systems Engineering

Any instance of system development, whether the domain is prosaic or not, can be generically described by a process as depicted in figure 1.

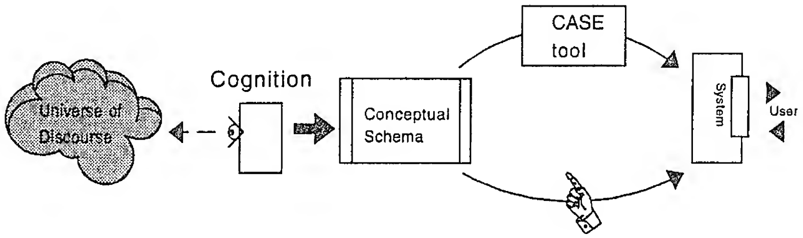


Fig. 1. Generic System Development Process

System developers and domain experts jointly observe (perceive) a universe of discourse (application domains) and apply a process of a methodological process of cognition[1] to these perceptions. The result of this activity is a so-called conceptual schema, which is a "partly formal, partly informal" representation of knowledge relevant to the application programs in the system to be built. The conceptual schema is used as a specification for the generation of this system; this generation can be automatic and/or manual. Especially in the case of prosaic domains, the automatic generation is important: the data structures and manipulation are usually very well understood, solutions are readily available and the applications are fairly shallow.

This is an ideal situation for CASE tools. Here a well-known weakness of CASE tools, as they are available on the market today, comes into play: when specifying and generating large numbers of independent updates transactions programs, the mutual interactions of such programs are hard to define. The natural place to specify the effects of such interactions would of course be in the conceptual schema; most formal conceptual schema techniques however support purely the specification of integrity constraints at the conceptual level. This leads in tearn to large systems which are difficult to maintain. Observe at this point that integrity constraints may be seen as a valid formal semantic mechanism: They effectively circumscribe the concepts to which they apply.

[1] Note that we use *cognition* in a limited sense, possibly different from the interpretations given in psychology and elsewhere in A.I. We define cognition as the addition/application of knowledge to perception.

To arrive at the specification of integrity constraints, indeed at *any* concepts laid down in the conceptual schema, the cognitive process is based on *agreement.*

Take the somewhat trivial example of storing an information system the fact that "John works for ABC". Since in this type of applications there will be in general many more instances of the same structure, registration of such a fact will proceed in two distinct phases: First data *types* are identified and specified in the conceptual schema, a *database* is generated, and in a second phase ususally after the system has been implemented with sufficient application programmes, the fact instance, or "tuple" (John ABC) is stored in this database by an application program. There needs to be agreement on the data types, on the integrity constraints on those data types, on the fact that John is an employee for instance, and that ABC is a company, and that actually John is working for ABC. To obtain agreement about the latter fact, it might be necessary to either read his employment contract, or to involve and consult an authority that will ascertain this fact to the system developer and/or domain expert, and so on...

The above reasoning is an obvious example of *reductionism*, a principle that underlays most of our formal treatment of knowledge. It has obvious disavantages, discussed at length in the literature, caused by a rapidly exploding level of complexity if one wants to support knowledge representation by computers. Nevertheless, we humans and other intelligent agents are capable of very quickly reaching agreements about such "mundane", "obvious" or "trivial" facts. Apart from certain references in common sense reasoning (Hobbs & Moore [13]) and new developments in Distributed A.I. [3] there seem to be available very few techniques, methods, and tools to handle the informality, or rather the required abstraction level in a representable formalism.

In this approach therefore, we view the cognition process as a *computation* (partially informal), which results in the (formal) agreement about a fact, rule, or simply object; this computation itself however is largely thrown away at the moment of agreement.

Naturally a lot of the semantics of the resulting agreed fact is determined by this cognitive computational process and it is precisely the lack of record about this process that causes the maintainability problem mentioned earlier. Documenting this specification clearly is only a partial remedy to this problem, if only because of the informality of it. There seem to be no adequate models available for registering the negotiation and the process that leads to the agreement, although recent research in dialog representation must be reported here (Bunt [5]). The missing knowledge is clearly of a linguistic nature, at least while the agreement process is taking place; methods that register more of this (natural language) linguistic information and some of the integrity constraints that emerge during the acquisition phase will therefore have an advantage (e.g. NIAM, [20] [21]).

Object-Role Models

At this point it is probably useful to dwell a little bit deeper on object-role models, a class of system specification notations and methodologies that have evolved since the late seventies to occupy today a limited but highly interesting niche in the specification of systems for, in particular, prosaic domains. Several varieties of object-role models exist: among them, NIAM is undoubtedly today the most widespread methodology, but it has spun off several enhancements and variations such as ORM ([12]), NORM (DeTroyer [6]), and others. In the methodological genealogy object-role models are probably preceeded by the functional model (Shipman [19]) and the binary model (Abrial [1]).

In the case of NIAM an extensive body of literature exists that refers to application, methodology as well as theoretical foundation. Good comprehensive text books are Wintraecken [21] and Nyssen & Halpin [18]. The NIAM model possesses an explicit distinction between labels (lexical objects) abstract entities (non-lexical objects), has a subtype mechanism and expresses relationships among object types through the use of so-called *facts* which have two (in some dialects several) *roles* that are expressed explicitly, based on verbs that occur in the sentences from which the facts are derived. This leads to quite verbose conceptual schemas; however by non-expert users this is perceived as an advantage because most conceptual schemas are meant to be written once, but consulted very often. In view of the discussion above and the re-engineering issues discussed in the next section, this advantage should be obvious, especially in the case of so-called prosaic domains. Additionally, NIAM explicitly supports integrity constraints of which some are represented in the diagram notation for the conceptual schemas. For those and other constraints several languages have been proposed over the years (Meersman [16], Mark & Roussopoulos [15], Halpin [11]). An example fragment of a NIAM conceptual schema is depicted in figure 2.

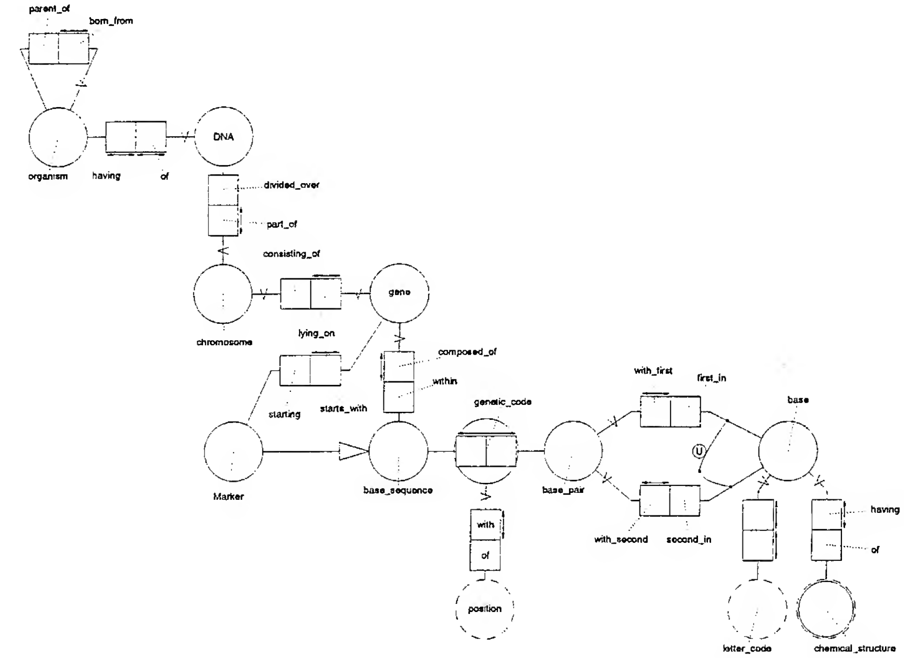


Fig. 2. Example fragment of a NIAM conceptual schema (for DNA)

The explicit role names and other linguistic information in NIAM schemas allows to re-generate in the form of stylized natural language all of the facts, relationships, rules and constraints that are represented in such a NIAM schema. While it does not constitute a formal support of the cognitive process in the strict sense, it has proved to be an easily accepted technique that is very useful in order to involve end-users and non-experts into the formal agreement process. Some CASE tools have implemented this feature as an enhancement to their "standard" tool box (RIDL* [9], InfoModeller [14]).

Re-Engineering and Interoperation of Information Systems

As indicated above, systems built for prosaic application domains are prime candidates for re-engineering and interoperation. They are difficult to replace and so making them interoperate with an other more recent system depends only on how the knowledge in the older system can be made availabe usefully to the new system, and vice versa, since we cannot usually modify the existing system. Furthermore, we have just seen that the specifications, or at least the semantics resulting from the original cognitive specification process, very often are lost. It becomes interesting therefore to study these mundane systems from the point of view of reconstruction of this lost knowledge. It should be obvious that the same techniques are applicable both for the re-engineering or reverse engineering of a legacy system as for the reconstruction of its conceptual schema semantics for the purpose of interoperation with another system.

Several approaches are available to us. If the domain experts and the original system developers are available, they could redo the cognition phase and record the required knowledge; or we can attempt to reconstruct the agreement and the missing semantics by induction from the (large) population of instances in the systems database; or we rebuild the schema from scratch and then devise away by which the old data and schema can be made available under the new specification.

In the first case, the existing conceptual schema to be enhanced or adapted is the focus of attention, in the second case it is primarily the existing population, and in the third case it is the Universe of Discourse itself that is studied again. Each approach evidently requires its own specific techniques. Since the first case assumes the presence of the original specifiers and experts, it is quite unlikely to occur in this form in practice: nevertheless, some techniques that reverse-engineer parts of specifications from e.g. the application programs have been devised and even have been applied with apparent commercial success; clearly this circumvents to a major extent the need to have the original developers available to a major extent. In the second case, new techniques of predicate induction are emerging from the area of machine learning (Muggleton [17], Flach [10], Banerji [2]) that show considerable promise in enhancing existing schemas with the "experience" that is represented in the large, old populations of such systems. Realistic approaches however, will have to deal with the problems that are presented by the fact that in many cases the information in older systems is heavyly encoded and therefore does not exhibit clear regular patterns to be discovered by the predicate invention algorithms. Imagine for example a data field in an employee record that can mean at the same time the date he left employment, his date of retirement or even even his day of death, the determination of which is to be found in another status field in the record. The constraints applicable to each situation (i.e. the predicates sought) will naturally be different in each case.

The third case, where we seek to make the old conceptual schema available by transformation to a newly designed one, is perhaps the most practical viable solution today. Good schema transformation formalisms and even tools exist; theories may be developed and are to a certain extent available (DeTroyer [7] [8]). However, little has been done on the subject of transforming the corresponding application semantics as they are represented in the programs. For instance, in the example above, a schema transformation may be fairly easily defined that relates the employee record above to a cleaned up one in which there are three fields for the respective dates. It is entirely another matter to automatically convert the old applications depending on the status code into new application programs working on the new schema, but with the old population.

Conclusion

As the primary targets for exercises of re-engineering and interoperability, so-called prosaic application domains and their associated systems provide interesting and practically useful case studies for a treatment of semantics from a methodological as well as cognitive point of view. (In this paper we treat cognition as the process of linking knowledge to perception inside an intelligent agent.)

In this context, semantics can be given a concrete interpretation in the form of constraints and rules on a data model. This allows among other things for an incremental definition of the meaning of conceptual schemas and specifications of information systems. The same treatment of semantics is relevant for re-engineering, reverse engineering and the establishment of interoperation since all these problems involve the reconstruction of lost knowledge when the "cognitive computation" was thrown away during the acquisition phase. No comprehensive or adequate methodologies seem to exist at this point that support this cognitive process in such a way that the computation can be recovered. We have briefly mentioned a few techniques that are relevant to the issue, such as induction and schema tranformation.

A fundamental aspect of this problem area is that all semantic processes occur in heterogeneous groups, who must agree formally on specifications that in general will serve as input to CASE tools. Refinements of existing and well-tried methods such as NIAM are necessary; in general system methodologies will need to evolve in order to accommodate more linguistic-type knowledge generated during the specification acquisition process.

References and Bibliography

1. Abrial, J.R.: "Data Semantics", in: Database Management Systems; Klimbe, Koffeman (eds.), North-Holland (1974)

2. Banerji, R.: "Learning in the Limit in a Growing Language", in: IJCAI Proceedings, Morgan-Kaufmann (1987)
3. Bond, A.H. & Gasser, L. (eds.): Readings in Distributed Artificial Intelligence, Morgan-Kaufmann (1988)0
4. Brodie, M.L.: "The Promise of Distributed Computing and the Challenges of Legacy Information Systems", in: Interoperable Database Systems; Hsiao, Neuhold, Sacks-Davis (eds.), North-Holland (1993)
5. Bunt, H.: "Representing Knowledge extracted from Natural Language Dialogue", in: Proceedings of PRICAI'94 Workshop on Knowledge Engineering and Applications-Beijing, Zhongzhi Shi (ed.) (1994)
6. De Troyer, O.: The OO-Binary Relationship Model: A Truly Object-Oriented Conceptual Model, in: Lecture Notes in Computer Science **498**: Advanced Information Systems Engineering; R. Andersen, J.A. Bubenko Jr., and A. Sølvberg, Springer-Verlag (1991)
7. De Troyer, O.: RIDL*, A Tool for the Computer-Assisted Engineering on Large Databases, proceedings ACM-SIGMOD (1989)
8. De Troyer, O.: On Data Schema Transformations, Ph.D. Thesis, ISBN 90-900591-3-x, Tilburg University (1993)
9. De Troyer, O., Meersman, R. and Verlinden, P.: RIDL* on the CRIS Case: A Workbench for NIAM, Proceedings of IFIP CRIS Working Conference; T.W. Olle et al. (ed.), Elsevier Science Publishers (1988)
10. Flach, P. A.: Predicate Invention in Inductive Data Engineering, in: Proceedings European Conference on Machine Learning ECML'93; P.B. Brazdil (ed.), Lecture Notes in Artificial Intelligence **667** , Springer Verlag (1993) 83-94
11. Halpin, T.: "A Logical Analysis of Information Systems", Ph.D. Thesis, University of Queensland (1989)
12. Halpin, T. & Meersman R. (eds.),: "Proceedings of the first Conference on Object-Role Modelling (ORM-1), Magnetic Island, (to appear 1994)
13. Hobbs, J. and Moore, R.C.: "Formal Theories of the Commonsense World", Ablex Publ. Co. (1985)
14. Infomodeller User Manuals, Asymetrix TM Corporation, Bellevue WA, U.S.A.
15. Mark, L. and Roussopoulos, N.: "Integration of Data, Schema and Meta-Schema in the context of Self-Documenting Data Models", in: ER approach to Software Engineering, North-Holland (1983)
16. Meersman, R.: "Towards Models for Practical Reasoning about Conceptual Database Design", in: Data and Knowledge (DS-2); Meersman, Sernadas (eds.), North-Holland (1986)
17. Muggleton, S.H. (ed.): Inductive Logic Programming, Academic Press (1992)
18. Nijssen, G.M. and Halpin, T.: "Conceptual Schema and Relational Database Design", Prentice Hall (1989)
19. Shipman, D.: "The Functional Data Model and the Data Language DAPLEX", in: ACM TODS vol 6 no.1 (March 1981)
20. Verheijen, G. and van Bekkum, J.: "NIAM: an Information Analysis Method", in: Proceedings of CRIS-1 Conference; Olle, Sol & Verrijn-Stuart (eds.), North-Holland (1982)
21. Wintraecken, J.-J.: "The NIAM Information Analysis Method-Theory and Practice", Kluwer Academic Publishers (1990)

Recognizing Credible Experts in Inaccurate Databases*

Hasan M. Jamil[†] **Fereidoon Sadri**
Department of Computer Science
Concordia University, Montreal, Canada
e-mail: {jamil, sadri}@cs.concordia.ca

Abstract: While the problem of incomplete data in databases has been extensively studied, a relatively unexplored form of uncertainty in databases, called *inaccurate data*, demands due attention. Inaccurate data results when data are contributed by various information agents with associated credibility. Though the data itself is *total* or *complete*, the reliability of the data now depends on the agents' credibility. Several issues of this form of data reliability has been reported recently where the credibility of agents were assumed to be *known*, *static* and *uniform* throughout the database. In this paper we address the issue of *credibility maintenance* of information agents and take the view that the agent credibility is *dynamic* and is a function of the database knowledge, the agent's *performance* relative to other agents, and the agent's expertise. We present a method to identify agents' field of expertise (called the *contexts*) and use agents' context dependent credibility to calculate the reliability of the *contextual data*.

Key words : approximate reasoning, information agent, contexts of data, data reliability, agent credibility, inaccurate data, uncertainty management.

1 Introduction

The issue of information reliability in *uncertain* databases is an active area of research. While the issue of *incomplete data* has been extensively studied, a different form of uncertainty in data and knowledge-bases, called the *inaccurate data*, is relatively unexplored. An inaccurate data is a *complete data* [6], the truth value of which is a probability that depends on the credibility of the contributing agents of the data. The credibility of an agent is the probability of the correctness of the agent. Sadri [7] proposed an extended relational model, called the *information source tracking* (IST) model, to represent and manipulate agent information along with the data and to calculate the data reliability as a separate process given the agents' credibility. The extended relational operations of IST model were shown to be *precise* (sound and complete). An algorithm was also presented that calculates the tuple reliability in the extended model.

In this paper, we take up the issue of agent credibility, which ultimately influences the data reliability, from yet another dimension. We define the credibility of an agent to be the estimate of the mean *approximation* of the data contributed by that

*This research was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada and the Fonds Pour Formation De Chercheurs Et L'Aide À La Recherche of Quebec.

[†]This author's research was additionally supported in part by grants from the Canadian Commonwealth Scholarship and Fellowship Plan and the University of Dhaka, Bangladesh. The author is on leave from the University of Dhaka, Bangladesh.

agent to the database *facts*. Database facts are *total* and *accurate* data that are *observed* to be *true*. The measure of approximation is based on the concept of *context-dependent similarity* in artificial intelligence and depends on the surrounding data from other agents in the same context. This approach allows us to model agents' context based credibility in a dynamic way and results in an improved data reliability. The credibility model we develop is also capable of recognizing an agent's expertise in different contexts.

The rest of the paper is organized as follows. In section 2, we present our data model which is an extension of Sadri's IST model [7], and discuss related issues. In section 3 we present the concept of agent credibility based on neighbourhood dependent similarity measure and statistical estimation. In section 4 we present a function for computing credibility of agents in a given context, and in section 5 an algorithm is outlined to calculate the reliability of answer tuples during query processing. Finally we conclude in section 6 and discuss further research issues. We omit discussions on related research and critical review of our work for the want of space. Details about agent vectors – its interpretations and manipulations, virtual agent vectors, extended relational operators, reliability algorithm, etc. are also left out for the same reason. Interested readers are referred to [4] for further readings.

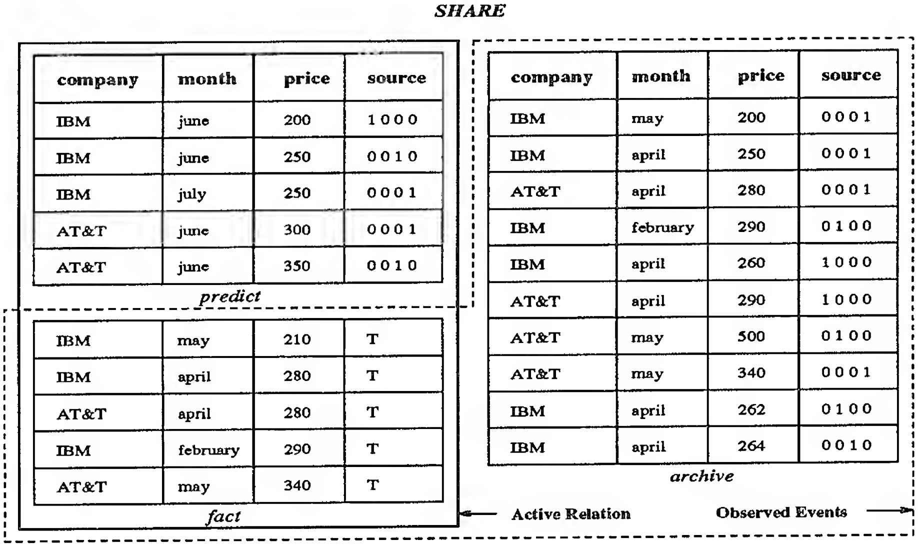
2 The Data Model

In this section we will introduce the data model and discuss query processing issues. We defer our discussion on reliability of answer tuples until after we formally define the concept of credibility in the next section. We will first illustrate the salient features of our model to give an intuitive overview, and then present the formal treatment.

2.1 The Extended IST Model

From the user's point of view, the database is a classical relational database. However,

- each relation scheme in the database has a special attribute called the *agent* or *source*, which is invisible to the users.
- during insertions and updates, users are asked to associate with each tuple a set of agents who contributed the tuple.
- in addition to producing answers to user queries, the system also produces the contributing agents of the answer tuples.
- for each answer tuple the system can calculate the probability of the answer being true. This probability is a function of the contributing agents' credibility.
- tuples in a relation are partitioned into three sets, *predict, fact,* and *archive*. The union of predict and fact set, called the *active relation*, takes part in query processing and database updates. The fact tuples are certain with a 100% reliability. The union of fact and archive sets is called the *observed set* or *observed events*. Tuples from the predict set migrate to the archive when *certain* tuples are added to the database fact. The set of tuples in the predict set that agree with the key of a fact tuple *t* are called the *twins* of *t* (may also be empty), and these are the tuples that migrate from predict to archive.
- each relation is optionally supplied with a *context*. A context is a set of regular attribute names that identifies agents' field of expertise in a relation.



SHARE

company	month	price	source
IBM	june	200	1 0 0 0
IBM	june	250	0 0 1 0
IBM	july	250	0 0 0 1
AT&T	june	300	0 0 0 1
AT&T	june	350	0 0 1 0

predict

company	month	price	source
IBM	may	210	T
IBM	april	280	T
AT&T	april	280	T
IBM	february	290	T
AT&T	may	340	T

fact

company	month	price	source
IBM	may	200	0 0 0 1
IBM	april	250	0 0 0 1
AT&T	april	280	0 0 0 1
IBM	february	290	0 1 0 0
IBM	april	260	1 0 0 0
AT&T	april	290	1 0 0 0
AT&T	may	500	0 1 0 0
AT&T	may	340	0 0 0 1
IBM	april	262	0 1 0 0
IBM	april	264	0 0 1 0

archive

Figure 1: The Share relation instance.

From the systems point of view, the database is an extended set of relations, with extended relational operations. If, however, the uncertainty component is turned off, the system behaves exactly like a classical relational model. We now formally define our extended model as follows:
Let $D = \cup_{i=1}^{k} D_i \cup D_I$ be a finite set of domains, where D_is are classical domains and D_I is a special domain consisting of a set of vectors of the form C^k where k is the number of agents in the database, $C = \{0, +1, -1, \top\}$, and includes two special vectors T and F corresponding to *true* and *false* propositions respectively. Let $A = \cup_{i=1}^{\infty} A_i \cup A_I$ be an infinite set of attribute names, where A_is are regular classical attributes, and A_I = *source* is a special attribute, called the information agent attribute, which draws its values from the domain D_I.

Definition 2.1 An *extended relation scheme* R is a triple $\langle$*Attributes, Constraints, Context*$\rangle$. Attributes is a set of attribute names $X = \{A_1, \ldots, A_n, A_I\}$, $n \geq 1$. Each attribute A_i has a domain of values D_i, $i = 1, \ldots, n$. The constraints is a set of conditions that the relation instances must satisfy. Context C is a set of attribute names such that $C \subset (X - A_I)$ and $K \not\subset C$, where K is a key of R. An *extended relation instance* (or just a relation) r on R is a set of tuples such that $r = r_{p(redict)} \cup r_{f(act)} \cup r_{a(rchive)} \subseteq D_1 \times \ldots \times D_n \times D_I$ such that r_p, r_f and r_a are pair-wise disjoint. □

An instance of a context is a tuple on C. A context C of a relation r partitions[1] the relation for all possible instances of C. Hence, every partition will have a predict, fact, and archive set, and all the tuples in a partition agree on one instance, e.g. tuple of values, c of C. An instance of a context also is a possible field of expertise for an agent. We will see later, that the expertise of an agent is a dynamic property

[1] A partition of r on c can be thought of as a SQL view select * from r group by c.

that changes with database activity. This approach to context[2] modeling allows us to capture the time dependent behavior of the expertise of an agent that is recognized dynamically and will be evident from the definition of credibility of agents.

Definition 2.2 Let t be a tuple in r. Then the projection $t[A_1, \ldots, A_n]$ is called a pure tuple or regular tuple while the projection $t[A_I]$ is called the agent vector. A tuple t is often represented by $p@a$, where $p = t[A_1, \ldots, A_n]$ denotes the pure component of the tuple, while $a = t[A_I]$ denotes the agent component of t. □

Definition 2.3 Let tuple $t = p@(a_1 \cdots a_n) \in r$, and S be the set of database agents. Let each agent $s \in S$ be associated with a natural number k from 1 to $|S| = n$, denoted s_k, and $t[C] = c$ be the context of the tuple $t \in r$. Let the k-th projection of the agent vector $a = t[A_I]$, denoted $t[A_I]^k$, represents the contribution of the k-th agent in tuple t in context c. Then $agree(s_k, t, c)$ represents the fact that the tuple t is confirmed by the agent s_k, i.e., $t[A_I]^k = a_k = +1$ where $t[C] = c$. □

We are now ready to formally define the concept of *tuple migration* from the predict set to the archive as follows. For theoretical reasons, it may be assumed that the three partitions of a relation are virtual. The following definition makes it clear that the partitions are only a mechanism to identify the tuples of active and observed parts of a relation.

Definition 2.4 Let r be a relation with key K. Then, the fact, archive and predict partitions of r are given by $r_f = \{t | t = p@a \in r \text{ and } a = T\}$, $r_a = \{t | t' \in r_f \text{ and } t \in (r - r_f) \text{ and } t'[K] = t[K]\}$, and $r_p = r - (r_f \cup r_a)$ respectively. □

Example 2.1 Consider the relation *Share* in Figure 1 with the scheme ⟨{*Company, Month, Price, Source*}, {*Company, Month* → *Price*}, {*Company*}⟩, where ⟨{*Company, Month, Price, Source*} are the attributes, {*Company, Month* → *Price*} is the set of constraints[3] (FDs), and {*Company*} is the context of the relation.

Consider the tuple t=<IBM, june, 200>@(1000) in *Share*. We say that <IBM, june, 200> is the pure tuple, and (1000) is the agent vector. Assume that there are four agents in the database, namely s_1 = forbes, s_2 = linden, s_3 = johnson, and s_4 = smith. Since the first projection of the agent vector is 1, we say that agent s_1 (i.e., forbes) have contributed to this tuple (i.e., *agree(forbes, t, IBM) = true*).

Now, suppose that we add a tuple <IBM, july, 290>@(T) to the relation *Share*. Obviously, this tuple is an observed fact and its truth is independent of any agent contribution. However, the introduction of this tuple potentially forces a set of predicted tuples to migrate to archive since they now become certain. For instance, the tuple <IBM, july, 250>@(0001) migrates to archive when <IBM, July, 290>@(T) is added to *Share*, since both share the same key, i.e., <IBM, july>. □

Definition 2.5 An extended database **D** is a quadruple ⟨**R**, I, S, $\mathcal{X}$⟩, where **R** is a set of extended relation schemes, I is a set of relation instances on schemes in **R**, S is a set of database information agents, and $\mathcal{X}$ is a function that assigns credibility to database agents in S. □

[2] In the rest of the paper we will use the term context c when we actually mean the instance of context C without any confusion.

[3] Notice that this *key constraint* is relaxed in our model in the following way. We allow several tuples in the predict and archive parts of a relation to have the same key as long as their agent vectors are distinct. The reason is obvious, i.e., we want to allow uncertain and possibly contradicting data to simultaneously co-exist in a relation until they are observed to be true or false.

2.2 Agent Vector Representation

Recall that in the basic IST, the credibility of an agent is uniform over the database, -i.e., an agent has a single credibility. So, to record the nature of contributions to a tuple $t \in r$ for all the agents in S, we needed a vector of size $|S|$. However, in our model an agent has several credibility across the database which varies from relation to relation, and from context to context. Hence we need to record an agent's confirmation, denial, etc. for each tuple in a relation in every context. The approach we take is that we create a ***virtual agent***[4] corresponding to each real agent for every relation and every context in a relation. For example, if there are two agents a_1 and a_2, three relations r_1, r_2 and r_3, two contexts c_{11} and c_{12} in relation r_1, one context c_{21} and c_{31} in r_2 and r_3 respectively, then we create a vector of size $2 \times (2+1+1)$, i.e., $(a_{ij} : i = 1, \ldots, 2,\ j = 1, \ldots, 4)$. In practice, however, for space efficiency, we may use a sparse matrix technique to represent the vectors, or only store a set of triples of the form <*agent, relation, context*>, and manipulate the matrix, or the sets of triples, which mimics the vectors. However, in general the size of the vector will be $|S| \times (|D_{C_{r_1}}| + \cdots + |D_{C_{r_d}}|)$, where $d = |I|$ is the number of relations in the database $\mathbf{D}$ and C_{r_i} is the context of relation r_i, $i = 1, \ldots, d$, and $D_{C_{r_i}}$ is the domain[5] corresponding to the context attribute in relation r_i.

Example 2.2 Let $\mathbf{D}$ be a database, s_1 and s_2 be the agents, and r_1, r_2 and r_3 be three relations in $\mathbf{D}$ each with only one context. This results in six virtual agents s_{ij}, $1 \le i \le 2$, $1 \le j \le 3$. Let t be a tuple as follows:

$$t@(\{\overbrace{(\underbrace{1\ 1\text{-}1}_{s_1}\quad \underbrace{0\ 1\ 1}_{s_2})}^{v_1}, \overbrace{(0\text{-}1\ 1\quad 1\text{-}1\ 0)}^{v_2}\})$$

Note that v_1 and v_2 are two vectors, and in each vector, s_1 and s_2 contribute in exactly three contexts. Now our goal will be to combine agents s_k's credibility in different contexts to compute its combined or overall credibility. □

Example 2.3 Consider the following two relations r_1 and r_2 with schemes $R_1 = \langle\{$*Company, Month, Price, Source*$\}$, $\{$*Company, Month* $\rightarrow$ *Price*$\}$, $\{$*Company*$\}\rangle$ and $R_2 = \langle\{$*Company, Address, Source*$\}$, $\{$*Company* $\rightarrow$ *Address*$\}$, $\rangle$ respectively.

Company	Month	Price	Source
IBM	June	200	100 100
AT&T	July	300	010 000

Company	Address	Source
IBM	New York	000 001
Apple	California	001 001

We have assumed that there are two database agents, say forbes and linden, and clearly there are two context values in r_1 (IBM and AT&T), and one context in r_2 (the key of r_2, or empty context). Hence, the length of each virtual agent vector is $2+1$, and the length of the agent vector is $2*(2+1)$. The vector (100 100) represents the fact that

[4] The concept of virtual agents was first developed in [9] to handle integrity constraints and associated inconsistencies.

[5] If $C \subset X$ and $|C| > 1$, then the domain of C is a relation defined as the Cartesian product of the domains corresponding to the attributes in C.

agent forbes and linden confirmed this tuple in the context of IBM in r_1, while the vector (010 000) indicates that only agent forbes confirmed this tuple in the context of AT&T in r_1.

Now consider the query, *"List the name and address of any company selling a share"*, i.e., $\Pi_{Company,Address}(r_1 \bowtie r_2)$. The answer to this query produces the following relation. Notice the new agent vector in the lone answer tuple as a result of extended relational operations[6]. The vector (100 101) carries the information that the agent forbes confirmed the tuple in relation r_1 in the context of IBM, while the agent linden confirmed it both in the context of IBM in r_1 and in the only context (empty) in r_2. □

Company	Address	Source
IBM	New York	100 101

Once we have chosen the agent vector representation as described above, the model becomes similar to basic IST with the only remaining difference in tuple reliability calculation, and all the algebra operations that are defined for basic IST still applies. It can be readily recognized that the elegance of IST is in the separation of reliability calculation from the vector manipulations. In that respect IST remains non-committal to any type of credibility assignment function to the agents, or to the virtual agents. Hence it is possible to use any suitable credibility assignment function specific to an application. This fact was utilized in [3] to assign a dynamic credibility to the agents in the database based on statistical sampling and the notion evidence. In the next section, we develop a method for assigning dynamic credibility to an agent based on the agent's expertise and the concept of similarity in artificial intelligence.

3 Credibility of Experts

In this section we concentrate on developing a formal definition of agent's expertise and the credibility. Intuitively, expertise of an agent is centered around an entity or concept about which the agent expresses her opinion or contributes information. The central purpose of a context in a relation is to partition the relation in terms of the domain values in the context that is treated as a possible field of expertise of agents for that relation. For example, in Figure 1, *Company* may be the context. Thus the values {IBM, AT&T} are the possible context values. We may find linden is more knowledgeable on IBM affairs than AT&T's simply because the credibility computed using tuples in the partition corresponding to IBM is higher than that of AT&T. Credibility on the other hand is intimately related to expertise. That is, we call an agent more credible on a subject than another if she is more expert on that subject than the other. The credibility we have in mind is based on a similarity measure which induces a total ordering on the credibility values for the contexts of an agent. Thus for a given agent α and a context c, if the credibility of α in c is higher than her credibility in another context c', then we say that the agent α is more knowledgeable on c than c'. In the following definitions we make these notions precise.

Definition 3.1 Let $t = p@a$ be a tuple in r_f with regular attributes X and key $K \subseteq X$. The function *twin* : $r_f \rightarrow 2^{r_a}$ is a mapping that associates with every tuple

[6] For a discussion on query processing using extended operators, readers are referred to [3, 4, 7, 8, 10].

t in *facts* of r, a set of *archive* tuples such that $twin(t) = \{t' | t' \in r,\ t'[K] = t[K]\}$. Let the *neighborhood* of t, denoted $\nu(t)$, be $\nu(t) = \{t\} \cup twin(t)$. □

We now define the concept of similarity between two tuples in the context, called the *neighbourhood*, of their twins. Our idea of similarity coincides with the concept of *context dependent similarity* in artificial intelligence [1]. In context dependent similarity, the presence of other objects in the observation space influences the degree of similarity between two objects. Hence a change in the objects in the observation space may change the degree of similarity. The fundamental assumption that is made for the measurement of similarity is that "two points in a relatively dense region of a space would have a smaller similarity measure than two points of equal inter point distance but located in a less dense region of space" [5]. This is the basis of our model of similarity too. There are many models for computing similarities between objects. Although we present here a function to measure the similarity of tuples in the context of their twins, we particularly do not commit to any such function. Instead, we describe some properties that a function should satisfy to be a candidate for use in our model, and leave it open for the users to choose their favorite function.

Definition 3.2 Let α be an agent, r be a relation, and c be a context instance. Then the *population* P of α in context c in r is a function such that $P(\alpha, c, r) \subseteq r$, and is given by $\{t | t \in r \wedge t[C] = c \wedge agree(\alpha, t, c)\}$. The *evidences* or the *samples* η of $P(\alpha, c, r)$ is a mapping $\eta(\alpha, c, r) \subseteq r_a$, and is given by $\{t | t \in r_a \wedge t[C] = c \wedge agree(\alpha, t, c)\}$. □

Observation 3.1 Given any agent α, and a context c in relation r, for every tuple t_i in the evidence set $\eta(\alpha, c, r)$, there exists a tuple $t \in r_f$ called the *image* of t_i, denoted $image(t_i)$, such that $t_i[K] = t[K]$ and $t_i[C] = t[C]$. □

Definition 3.3 Let $y \in r_a$ and $x = image(y) \in r_f$. Then the *similarity* of x and y with respect to the neighbourhood of x is a function $\psi_{\nu(x)}$ such that $\psi_{\nu(x)}(x, y) \in [0, 1]$, and the complement of similarity, called the *dissimilarity* $\delta_{\nu(x)}$, is given by $1 - \psi_{\nu(x)}(x, y)$. □

We require that the similarity function to have the following properties:

- Identity: $\psi_{\nu(x)}(x, x) = 1$.
- Monotonicity: Let $\preceq_D$ be an ordering on the domain elements of D (called the inter element distance[7]). If $d_1 \preceq d_2 \preceq d_3$ holds, then for a fixed context c, $\psi_{\nu(c)}(d_1, d_2) \geq \psi_{\nu(c)}(d_1, d_3)$.

Definition 3.4 For a given agent α, a relation r, and a context c, the *mean similarity* of tuples $\eta(\alpha, c, r)$, denoted $\mu(\alpha, c, r)$, is the mean of the individual similarities for each tuple $t \in \eta(\alpha, c, r)$ with its image, and is given by

$$\mu(\alpha, c, r) = \frac{\sum_{t \in \eta(\alpha, c, r)} \psi_{\nu(image(t))}(image(t), t)}{|\eta(\alpha, c, r)|}$$

which has a range $[0, 1]$. □

[7] Elements of D can be thought of as tuples in general.

We are now ready to define *credibility of agents*. The idea is to use the mean similarity measure of the observed (archive) tuples in a context for an agent and to estimate[8] the population mean. The observed tuples may be viewed as random samples. The goal of this estimation is to compute the *nearness* of the predicted data contributed by an agent to the database facts that are certain. The larger the value of estimation, the more reliable the data contributed be the agent will be.

Definition 3.5 The *credibility* $\mathcal{X}$ of an agent α in a context c in r is a function such that $\mathcal{X}(\alpha, c, r) \in [0, 1]$. It is an estimate [2] of the mean similarity of the population $P(\alpha, c, r)$, and is given by

$$\mathcal{X}(\alpha, c, r) = \begin{cases} \mu(\alpha, c, r) \pm z_a \frac{s}{\sqrt{n}} & \text{if } n \geq 30 \\ \mu(\alpha, c, r) \pm t_{(n-1,a)} \frac{s}{\sqrt{n}} & \text{otherwise} \end{cases}$$

where, $n = |\eta(\alpha, c, r)|$ is the sample size, s is the sample standard deviation, and z_a and $t_{(n-1,a)}$ are the critical values at confidence level a obtained from standard charts for normal and t distributions respectively [2]. □

Theorem 3.1 Let $N = |P(\alpha, c, r)|$ be the population size of an agent α in a context c in r, and $n = |\eta(\alpha, c, r)|$ be the sample size of the observed evidences of α. If $\mu(\alpha, c, r)$ is the mean similarity of the observed evidences, s is the sample standard deviation, and $n \geq 30$, then the computed credibility $\mathcal{X}(\alpha, c, r)$ is correct within $\pm z_a \frac{s}{\sqrt{n}}$ with confidence a. □

4 Computing Neighbourhood Dependent Similarity

In this section, we will present a function to compute similarity which we adopt from [1], and discuss the function only briefly. Interested readers may be referred to [1] for details. This function satisfies the required properties of our credibility function. In particular it satisfies a special case of the monotonicity property as follows. For any tuple x, y and z, $\{k | x_k \neq z_k\} \subset \{k | x_k \neq y_k\} \cup \{k | y_k \neq z_k\}$, and any neighbourhood n, $\delta_n(x, z) \leq \delta_n(x, y) + \delta_n(y, z)$ holds. Cheng's [1] formula for measuring neighbourhood dependent similarity[9] of two objects is

$$\psi_{\nu(x)}(x, y) = 1 - \frac{\sum_{k, x_k \neq y_k} (\log p_k + \log(1 - p_k))}{\sum_k (\log p_k + \log(1 - p_k))}$$

where p_k is the smallest among the probabilities of the values of the kth attribute of the tuples in $\nu(x)$. The formula follows from the idea of ***attribute weighting, differential weighting*** and ***metrization of differential weighting*** [1].

Definition 4.1 Let the neighbourhood of $t \in r_f$ be $\nu(t)$. Let the frequency f_i of a distinct value v_i be the count of number of agents confirming the tuple where v_i appears in the k-th attribute of all the tuples. The ***least probability*** of the k-th attribute values of the tuples in $\nu(t)$, denoted $p_k(t_k)$, is the ratio of the minimum frequency to the sum of the frequencies, i.e., $min(\{f_1, \ldots, f_n\}) / \Sigma_i f_i$, where n is the number of distinct values in the projection of the kth attribute of $\nu(t)$. □

[8] In [3], an estimate based on proportions was used that lacked accuracy since only binary match of tuples was considered. The current approach is intuitive, and the use of the concept of similarity of tuples significantly improves the accuracy of the agent credibility estimate.

[9] We will use the term neighbourhood dependent similarity to avoid confusion with our concept of contexts of a relation.

5 Algorithm for Tuple Reliability Calculation

We develop an outline of the reliability calculation algorithm for answer tuples in this section. This algorithm is based on the algorithms presented in [7, 8]. For simplicity, we will assume point credibility for agents instead of ranges in this paper. The basic idea is to calculate the combined credibility of an agent in all the contexts based on the notion of *positive correlation* and use the algorithm for basic IST.

In basic IST [7], the agent vectors are first converted into the *disjunctive normal form* [8]. Then the reliability of tuple t is given by $rel = \sum_{i=1}^{p} reliability(v_i)$, where $v_1, \ldots, v_p$ are the normalized vectors of t. We define the function *combined_credibility* as shown in the figure below to compute the combined credibility of an agent based on positive correlation. In this function, we used the fact that given a virtual agent vector v, the *degree of denial* or *confirmation* in a context c_i is the estimated credibility of the agent in c_i with which she denies or confirms the tuple[10].

```
Function combined_credibility;
Input : agent vector $u$ of length $q$;
Output : combined credibility of agent $s_i$
begin
    $difference = min$(set of degree of
        confirmations) $- max$(set of degree of
        denials);
    if $difference > 0$ then
        return $credibility = difference$
    else
        return $credibility = 0$;
end;
```

```
Function reliability;
Input : tuple $t@v$;
Output : reliability of tuple $t@v$;
begin
    Extract virtual agent vectors $u_i$
        for each agent $s_i$ in $v$.
    $reliability = \prod_{i=1}^{k} credibility(u_i)$;
    return $reliability$;
end;
```

It is now very easy to develop a complete algorithm for tuple reliability once we have defined the functions for combined credibility, and reliability as in the above figure, and the algorithm for converting vectors to disjunctive normal form is known from [8], hence is left as an exercise. However, a discussion on the complexity of this algorithm, and the basis for the combined credibility function can be found in [4].

6 Conclusions

We have presented a model for representing inaccurate data in relational databases and introduced the concept of context in the relation schemes to model contextual data[11]. We have proposed the concept of dynamic agent credibility based on similarity measures of tuples, and statistical estimation. We have also presented an algorithm to compute the reliability of answer tuples in our model. Since credibility is computed during query processing, the time dependent behaviour of the agent credibility is captured in a clean way. We have shown that the concept of contexts are useful for recognizing the expertise of agents.

Our approach to uncertainty management has several advantages over contemporary ways of uncertainty management. The elegance of IST based systems are that no probability values are associated with the data, and the method of assigning credibility to agents is not fixed. Hence, tailoring and updating of credibility assignment

[10] Note that in the above algorithm, the set of confirmations always contains degree 1, and similarly, the set of denials always contains 0.

[11] The idea of contexts in IST was first introduced in [10] where a relation was implicitly treated as a context.

function is possible, either for static assignment [7, 10] or for dynamic assignment as in this paper. Due to the separation of reliability calculation and credibility assignment to agents, data reliability can be changed only by changing the agent credibility without changing the database in a nice and clean way – a substantial improvement compared to the contemporary quantitative models. This fact may be exploited for knowledge representation and non-monotonic reasoning applications. This model can also be viewed as an adaptive system, or as a cognitive model, where decisions must be made based on the *best* available knowledge from multiple knowledge-bases or agents.

Although, the present proposal extends and strengthens the basic IST in several ways, further extensions are possible. One possible improvement may be to extend the relation schemes by introducing an ordering relation (a concept hierarchy) on each attribute domain elements that are taxonomical in nature. A suitable similarity function may be devised that is capable of exploiting the domain orderings to produce a more accurate measure, and consequently improving the credibility measure. Notice that the current function only allows binary match of attribute values.

Another observation is that the number of archive tuples in a relation will grow much faster than the fact tuples, whereas the predict tuples are the dominant part of the relations. It might be possible to reduce the size of the archive under some suitable conditions without compromising the accuracy of the credibility measure too much. Again an approach based on statistical estimation and *tuple age* seems promising and interesting[12]. In the same token it might also be interesting to study *update* issues in our model. An interesting twist that is worth investigating is to enable the credibility maintenance system to discover the contexts for the agents. These are some of the issues we seek to investigate in our future research.

References

[1] Cheng, Y.; "Context-Dependent Similarity"; *Uncertainty in Artificial Intelligence 6*; Elsevier Science Publishers; 1991; pp 41-47.

[2] Cochran, W. G.; "Sampling Techniques"; *John Wiley & Sons, NY*; 1977.

[3] Jamil, H. M. and Sadri, F.; "Trusting an Information Agent"; *In Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, Banff, Alberta;* October, 1993; (Also to appear in Springer-Verlag).

[4] Jamil, H. M. and Sadri, F.; "Recognizing Credible Experts in Inaccurate Databases"; *Technical Report, Department of Computer Science, Concordia University, Montreal;* July, 1994.

[5] Krumhansl, C. L.; "Concerning the Applicability of Geometric Models to Similarity Data: the Interrelationship Between Similarity and Spatial Density"; *Psychological Review*, 85; 1978; pp 445-463.

[6] Maier, D.; "The Theory of Relational Databases"; *Computer Science Press*; 1983.

[7] Sadri, F.; "Reliability of Answers to Queries in Relational Databases"; *IEEE Transactions on Knowledge and Data Engineering*; Vol 3, No 2, pp 245-251; 1991.

[8] Sadri, F.; "Modeling Uncertainty in Databases"; *IEEE 7th International Conference on Data Engineering*; pp 122-131; 1991.

[9] Sadri, F.; "Integrity Constraints in the Information Source Tracking Method."; *To appear in IEEE Transactions on Knowledge and Data Engineering.*

[10] Shiri, N. and Jamil, H. M.; "Uncertainty as a Function of Expertise"; *Proceedings of the Workshop on Incompleteness and Uncertainty in Information Systems, Montreal, Quebec*; October, 1993; (Also to appear in Springer-Verlag).

[12] If we would like to remove tuples from the archive set, it might be a good idea to remove the oldest possible tuples, the aged ones, that do not account for the most up to date behaviour of the agent performance.

FUZZY LOGIC OR LUKASIEWICZ LOGIC: A CLARIFICATION

Sukhamay Kundu and Jianhua Chen

Computer Science Department
Louisiana State University
Baton Rouge, LA 70803-4020
E-mail: {kundu, jianhua}@bit.csc.lsu.edu

ABSTRACT

Pavelka [10] had shown in 1979 that the only natural way of formalizing fuzzy logic for truth-value in the unit interval [0, 1] is by using Lukasiewicz's implication operator $a \to b = \min\{1, 1 - a + b\}$ or some isomorphic form of it. A considerable number of other papers around the same time had attempted to formulate alternative definitions for $a \to b$ by giving intuitive justifications for them. There continues to be some confusion, however, even today about the right notion of fuzzy logic. Much of this has its origin in the use of improper "and" ("or") and the "not" operations and a misunderstanding of some of the key differences between "proofs" or inferencing in fuzzy logic and those in Lukasiewicz's logic. We point out the need for defining the strong conjunction operator "⊗" in connection with fuzzy Modus-ponens rule and why we do not need the fuzzy Syllogism rule. We also point out the shortcomings of many of the alternative definitions of $a \to b$, which indicate further support for Pavelka's result. We hope that these discussions help to clarify the misconceptions about fuzzy logic.

1. INTRODUCTION

Since the introduction of fuzzy sets by Zadeh [17], there have been numerous studies on developing an appropriate notion of fuzzy logic in order to handle the fuzzy reasoning tasks. The notion of *grade of membership* of an element x in a universe U with respect to a fuzzy subset P over U is regarded as the *degree of truth* of the statement "x is P". This forms the connection between fuzzy sets and the many-valued logic over the truth-value set [0,1]. The notion of fuzzy logic is a generalization of many-valued logic in that in the former we infer new facts along with their truth-values whereas in many-valued logic one infers only those facts that are absolutely true (have truth-value 1). For simplicity, we restrict the discussion to the propositional case.

The idea of considering intermediate truth values rather than just the two-valued truth set {0, 1} has been used extensively by Lukasiewicz and others since the 1920's [8]. Among the various many-valued logics, the Lukasiewicz logic $L_{\aleph}$ with [0,1] as the truth-value set are considered to be one of the most attractive candidates for fuzzy logic. The language of $L_{\aleph}$ has two primitive logical connectives $\{\to, \neg\}$, where "$\to$" is the Lukasiewicz implication given by $a \to b = \min\{1, 1 - a + b\}$ $(a, b \in [0,1])$ and $\neg a = 1 - a$ is the ordinary negation operation. The Lukasiewicz logic has been studied in numerous papers on fuzzy logic, with proposed extensions/generalizations for handling fuzzy reasoning [6-7, 10, 12]. Much of the focus in these studies is on alternative definitions of the operator "$\to$". Smets and Margrez [12] took an axiomatic approach for defining "$\to$" and showed that under a reasonable set of postulates it coincides with Lukasiewicz's implication, upto an isomorphism. One of the most significant work on the relations between fuzzy logic and the $L_{\aleph}$ is Pavelka's work [12], where he showed that under some very natural generalizations of the classical logic, the only structures that possess the completeness property are those which are isomorphic to $L_{\aleph}$. Lukasiewicz logic and its isomorphs are the only ones that are axiomatizable. A logic defined by a semantics is axiomatizable if there is a set of axioms and inference rules which are sound and complete with respect to the semantics. The best choice for the implication operator "$\to$" is therefore the Lukasiewicz implication. Although Pavelka's work has been recognized by some [4, 14], it has gone unnoticed to a large extent.

After almost 30 years of research on fuzzy sets and fuzzy logic, there still continues to be some confusions about the right notion of fuzzy logic. It is the purpose of this paper to clarify some of the confusions and misconceptions. One common confusion has to do with the choice of appropriate definition for "$\rightarrow$"; this is evidenced by the proliferation of different "$\rightarrow$" in the literature (cf. Section 4.2) and the fact that these are often justified only on some intuitive grounds, ignoring the basic logical issues involved. Another major source of confusion has to do with the fact that there are two different "or" operations (and similarly two "and" operations), whose different roles must be recognized. The ordinary "or" operation $A \vee B$ is defined in L_K by the formula $(A \rightarrow B) \rightarrow B$, which is equivalent in Boolean-logic to $(A \wedge \neg B) \vee B = A \vee B$. In Boolean-logic, we also have $\neg A \rightarrow B = A \vee B$; in L_K the formula $\neg A \rightarrow B$ is taken to be the definition for the strong-or operation, which is denoted by $A \oplus B$. The truth values of these different "or" operations and their corresponding "and" operations are given by

$$\begin{array}{ll} \tau(A \vee B) = \max\{\tau(A), \tau(B)\} & \tau(A \wedge B) = \min\{\tau(A), \tau(B)\} \\ \tau(A \oplus B) = \min\{1, \tau(A) + \tau(B)\} \geq \tau(A \vee B) & \tau(A \otimes B) = \max\{0, \tau(A) + \tau(B) - 1\} \leq \tau(A \wedge B\} \end{array}$$

Another factor which is often overlooked is that when these operations are generalized further, one must use a corresponding generalization of the negation operator $\neg a = 1 - a$ as well (cf. Section 4.1).

We will briefly recall in Section 2 some basic definitions and concepts in L_K and in fuzzy logic. Section 3 focuses on the difference between the inferencing in fuzzy logic and L_K. In Section 4, we review Pavelka's results and discuss the issue of defining implication in a fuzzy logic. We summarize our conclusions in Section 5.

2. BASIC CONCEPTS

Historically, one of the main motivations behind the development of L_K was to create a logical system which is free of some of the paradoxes in the two valued Boolean-logic. For example, the formula $\phi = (p \rightarrow \neg p) \rightarrow \neg p$, which is a tautology (has truth value 1 for each value of $\tau(p)$) in Boolean logic, is no longer a tautology in L_K; for $\tau(p) = 1/2$ it has the truth value only 1/2. This is not to say that there are no unusual tautologies in L_K, although there are fewer of them in L_K than in Boolean logic. The formula $\phi = (p \rightarrow \neg p) \vee (\neg p \rightarrow p)$ is a tautology in both Boolean logic and in L_K; the unusualness of it is that it means the statement "if John is weak, then he is strong or if John is strong, then he is weak" is always true.

The logic L_K has two primitive logical connectives: "$\neg$" and "$\rightarrow$". The other connectives $\wedge$, $\vee$, $\leftrightarrow$ are defined as syntactic abbreviations for certain formulas involving only "$\neg$" and "$\rightarrow$". For example, we write $A \vee B$ for $(A \rightarrow B) \rightarrow B$. Well-formed formulas are defined in the usual way. We can view the class of all (well-formed) formulas F as a free-algebra like structure, over the propositional symbols, the constant symbols T (true) and $\perp$ (false), and the operator symbols "$\rightarrow$" and "$\neg$". In particular, no two formulas are considered to be the same unless they are exactly identical symbol by symbol. In this sense, we do not regard A and $\neg\neg A$ to be the same or $A \rightarrow B$ to be the same as $\neg B \rightarrow \neg A$. An interpretation I is a truth-value function $\tau(A)$ from the set of all propositions to [0,1]. The truth-value function is extended to F by means of homomorphism, i.e., we define

$$\begin{array}{ll} \text{(i)} & \tau(\neg A) = \neg \tau(A) \\ \text{(ii)} & \tau(A \rightarrow B) = \tau(A) \rightarrow \tau(B) \end{array}$$

where the "$\neg$" and "$\rightarrow$" on the left hand side of (i)-(ii) are operators in F and those on the right hand side are Lukasiewicz's operators on [0,1]. Thus, $\tau(A \rightarrow B) = \min\{1, 1 - a + b\} = \min\{1, 1 - (a - b)\}$, where $\tau(A) = a$ and $\tau(B) = b$, and $\tau(\neg A) = 1 - a$. We say that a formula ϕ is a *tautology* in L_K if ϕ has the truth value 1 in every interpretation; this is denoted by $\models \phi$. This constitutes the *semantical* definition of truth in L_K. Note that this definition of L_K implies the following important principle:

The Substitution Rule:
If we replace all occurrences of a proposition in a tautology (i.e., a semantically true formula) by an arbitrary formula, then the result is also a tautology.

Clearly, the notion of a tautology is intimately connected with the way the operations $\neg a$ and $a \rightarrow b$ are defined on the truth values. (Note that $\tau(A \rightarrow \perp) = \tau(\neg A)$.) In particular, one may not adopt any intuitive definitions for them without detailed examination of their effects on the resulting notion of true formulas (e.g., their structural properties).

The semantic notion of truth does not, in itself, make the relationships among tautologies (other than those related by substitution) very clear. To achieve this, one uses an equivalent *syntactic* definition of truth. A syntactic definition of truth has two parts:

(1) A set of *axioms*; this is simply a subset of well-formed formulas which are "declared" to be true.

(2) A set of *inference* rules; they are used for "deriving" new formulas from other formulas.

A formula ϕ is then said to be syntactically true (a theorem) if it can be derived in zero or more steps starting from the axioms; this is denoted by $\vdash \phi$. Each axiom is clearly a theorem. The substitution rule is always taken to be one such inference rule, in keeping with the corresponding property of semantic notion of truth. Note that the syntactic notion of truth does not involve computation of truth-values of formulas and thus does not require definitions of the truth functions $a \rightarrow b$ and $\neg a$.

We say that a semantic definition is *axiomatizable* if there is a suitable set of axioms and a set of inference rules such that the notion of semantic truth coincides with that of the syntactic truth, i.e., $\models \phi$ if and only if $\vdash \phi$ for every ϕ. In that case, each axiom must be a semantically true formula and can be therefore regarded as a constraint on the definitions for $a \rightarrow b$ and $\neg a$. A similar remark holds for each inference rule in that the derived formula must have the truth-value 1 if all the starting formulas have the truth-value 1. An axiomatization of $L_{\aleph}$, which uses five axioms and Modus-ponens inference rule (together with the substitution rule), is given in [11]. In this logic, we have $\models A \rightarrow ((A \rightarrow B) \rightarrow B)$; this is tantamount to Modus-ponens inference rule.

There are some key differences between inferencing in many-valued logic and that in the two valued Boolean logic. In the latter, we have $\phi \vdash \psi$ (i.e., starting from ϕ and the axioms in the logic, we can derive ψ) if and only if $\vdash \phi \rightarrow \psi$. This is no longer true in many-valued logic. For example, one can show that in $L_{\aleph}$: $A \vdash [A \rightarrow (A \rightarrow B)] \rightarrow B$, but $\not\vdash A \rightarrow ((A \rightarrow (A \rightarrow B)) \rightarrow B)$. For $a = 2/3$ and $b = 1/3$, the formula $A \rightarrow ((A \rightarrow (A \rightarrow B)) \rightarrow B)$ has truth-value only 2/3 and hence is not a tautology.

There are two approaches in defining the fuzzy implication operator "→" on [0,1]. One type of implication, called *R-implication*, is defined via the *residuation* operation in a *residuated lattice* over [0,1]. A residuated lattice over [0,1] is a lattice $L = \langle [0,1], \leq, \otimes, \rightarrow \rangle = \langle [0,1], \leq, \wedge, \vee, \otimes, \rightarrow \rangle$, where "$\wedge$" and "$\vee$" are the usual lattice meet and join operations defined by the partial order relation "$\leq$", $\langle [0,1], \otimes, 1 \rangle$ is a commutative monoid, and the pair of operators $\langle \otimes, \rightarrow \rangle$ forms an *adjoint couple* on [0,1]. An adjoint couple $\langle \otimes, \rightarrow \rangle$ is defined by [10]:

(1) $\otimes$ is isotone (monotone non-decreasing) in each of the arguments a and b,

(2) $a \rightarrow b$ is antitone (monotone non-increasing) in the first argument a and isotone in the second argument b, and

(3) $a \otimes b \leq c$ if and only if $a \leq b \rightarrow c$ for all $a, b, c \in L$.

It is not difficult to see that for an R-implication, $a \rightarrow b = 1$ if and only if $a \leq b$. The operators "→" and "$\otimes$" in $L_{\aleph}$ given in Section 1 form an adjoint couple.

Another type of implication is called *S-implication*. In this approach, one first defines a disjunction operator $\cup$ and then defines $a \rightarrow b$ to be $(\neg a \cup b)$. The function $s(a, b) = a \cup b$ is called an s-norm.

3. INFERENCE IN MANY-VALUED LOGIC AND FUZZY LOGIC

Inferencing in fuzzy logic not only derives a new formula from one or more other formulas but it also derives a truth-value for the new formula from those of the starting formulas. This is unlike the inferencing in many-valued logic and in classical binary logic which does not involve any truth-value as such. We can alternatively say that in many-valued logic, we do not infer formulas whose truth value is different from 1 and nor can we use such formulas as the starting point. In the fuzzy logic based on Lukasiewicz's $a \rightarrow b$, Modus-ponens inference rule takes the following form (see Example 1 below):

$$\frac{\langle A,\ p\rangle,\ \langle A \rightarrow B,\ q\rangle}{\langle B,\ r\rangle},\ \text{where } r = \max\{0,\ p+q-1\}.$$

Here we infer the formula B from $\{A,\ A \rightarrow B\}$ together with its truth-value r from the truth values p for A and q for $A \rightarrow B$. Note that if we do not know the truth-value for A or $A \rightarrow B$, then we cannot apply the fuzzy Modus-ponens rule. This shows that in fuzzy logic we need to represent formulas with attached truth-values. In Pavelka's formal definition of the fuzzy logic one can express the truth-value of a formula in the language itself (see next Section). The following examples show the fuzzy Modus-ponens and fuzzy Syllogism inference rules in the fuzzy $L_{\aleph}$.

Example 1. In fuzzy Modus-ponens inference rule, we wish to determine the truth-value $\tau(B) = r$ from $\tau(A) = p$ and $\tau(A \rightarrow B) = q = \min\{1,\ 1-p+r\} \geq 1-p$. This is done as follows using the fact $p \otimes q = \max\{0,\ p+q-1\}$. There are two cases to consider:

(1) $q = 1$. In this case, we can only say that $r \geq p = p \otimes q$ and cannot recover the exact value of r. This is not surprising because all values of $r \geq p$ give the same value $q = 1$.

(2) $q < 1$ (and $1-p \leq q$). In this case, we have $0 \leq r = q-(1-p) = p+q-1 = p \otimes q < p$.

We summarize the above observations in the form

$$\frac{\langle A,\ p\rangle,\ \langle A \rightarrow B,\ q\rangle}{\langle B,\ r\rangle},\ \text{where} \begin{cases} r = p \otimes q, \text{ if } q < 1 \text{ (and } 1-p \leq q) \\ r \geq p \otimes q, \text{ if } q = 1 \end{cases}$$

Since $p \otimes q$ is isotonic (monotone increasing) in both p and q, we can restate the above observation (at the cost of being less exact for the case $q < 1$) in a slightly simplified form as shown below, where we explicitly indicate that each truth value is a lower bound. The bound $r = p \otimes q$ is clearly tight.

$$\frac{\langle A,\ \geq p\rangle,\ \langle A \rightarrow B,\ \geq q\rangle}{\langle B,\ \geq p \otimes q\rangle}\ \text{(assuming } 1-p \leq q).\ ♣$$

One may wonder how would a fuzzy Syllogism inference rule look like. As the following example shows such an inference rule would be unnecessary if we have fuzzy Modus-ponens rule.

Table 1. Derivation of a fuzzy Syllogism inference rule.

	$a \leq c$			$c < a$		
	$b < a$	$a \leq b \leq c$	$c < b$	$b < c$	$c \leq b \leq a$	$a < b$
$p = \min\{1,\ 1-a+b\}$ $q = \min\{1,\ 1-b+c\}$	$1-a+b$ 1	1 1	1 $1-b+c$	$1-a+b$ 1	$1-a+b$ $1-b+c$	1 $1-b+c$
$r = \min\{1,\ 1-a+c\}$	1			$1-a+c$		
$p \otimes q$	$1-a+b$ $< r$	1 $= r$	$1-b+c$ $< r$	$1-a+b$ $< r$	$1-a+c$ $= r$	$1-b+c$ $< r$

Example 2. In fuzzy Syllogism inference rule, we wish to determine $r = \tau(A \to C)$ from $\tau(A \to B) = p$ and $\tau(B \to C) = q$. The following table summarizes the different situations in terms of $\tau(A) = a$, $\tau(B) = b$, and $\tau(C) = c$; we also show the corresponding values of p, q, r, and $p \otimes q$.

Notice that for the case $p \otimes q < r$, the values of $p \otimes q$ and r can be arbitrarily close depending on how close b is to a or c. The condition $1 - p \leq q$ holds as in Modus-ponens. We can summarize the information in Table 1 (at the cost of being less exact) as follows.

$$\frac{\langle A \to B, \geq p \rangle, \langle B \to C, \geq q \rangle}{\langle A \to C, \geq p \otimes q \rangle} \text{ (assuming } 1 - p \leq q).$$

We can derive the above rule by applying Modus-ponens twice. The first application involves the formulas $A \to B$ and $(A \to B) \to ((B \to C) \to (A \to C))$, where the latter has truth value 1 (it is a theorem in $L_\aleph$). We get

$$\frac{\langle A \to B, \geq p \rangle, \langle (A \to B) \to ((B \to C) \to (A \to C)), \geq 1 \rangle}{\langle (B \to C) \to (A \to C), \geq p \rangle}, \text{ since } p \otimes 1 = p.$$

Next, since $1 - p \leq q$ if and only if $1 - q \leq p$ and $p \otimes q$ is isotone in both p and q, we have

$$\frac{\langle B \to C, \geq q \rangle, \langle (B \to C) \to (A \to C), \geq p \rangle}{\langle A \to C, \geq p \otimes q \rangle}, \text{ by } q \otimes p = p \otimes q.$$

This gives the same bound $p \otimes q$ for $A \to C$ as obtained from Table 1.

Why does the above derivation of fuzzy Syllogism rule from fuzzy Modus-ponens rule fail to indicate that $p \otimes q$ may be strictly less than $\tau(A \to C) = r$ in many cases (where one of p and q were equal to 1) as had been observed in Table 1? We may say that it is hidden in the first application of Modus-ponens where one of the facts have truth-value 1. We remark that it is the analysis in Table 1 that shows in a more definitive way that there is no simple description of the exact truth value $\tau(A \to C)$ in terms of $\tau(A \to B)$ and $\tau(B \to C)$ other than in the form of "≥" bounds. The derivation via Modus-ponens, on the other hand, only suggests that perhaps that is the case. ♣

4. THE FUZZY IMPLICATION OPERATOR

The issue of finding a suitable definition for the fuzzy implication operator "→" has been studied by many researchers. Since the main purpose of defining $\tau(A \to B)$ is to carry out inferencing using facts and rules, the R-implications offer a good choice. Pavelka [10] showed that the Lukasiewicz implication and its isomorphic forms are the only suitable choice. In this section, we review Pavelka's results and point out shortcomings of some other definitions of "→".

4.1. Pavelka's Formalization of Fuzzy Logic

Pavelka's work concerns only with R-implications and uses an abstract truth value set L which is a residuated lattice. Recall that the interval [0,1] with → and ⊗ as defined in $L_\aleph$ is a residuated lattice. He formally defined the notions of L-semantics and L-syntax. He introduced a distinct atom $\bar{\alpha}$ for each $\alpha \in L$ in his language. Each interpretation assigns the truth value α to the atom $\bar{\alpha}$. This allows him to construct a fuzzy logic by considering the formalisms of multivalued logic where the inference rules deal with only formulas of truth value 1. The formula $\bar{\alpha} \to \phi$ is now a theorem (has truth value 1) if and only if ϕ has truth value $\geq \alpha$ in all interpretations; on the other hand, ϕ has truth value α in all interpretations if and only if $\alpha \leftrightarrow \phi$ (i.e., $(\alpha \to \phi) \wedge (\phi \to \alpha)$) is a theorem. The previous works by Goguen [6] and Gottwald [7] do not consider the atoms $\bar{\alpha}$ in the language and thus they lack the ability to explicitly reason about the truth-values syntactically.

An L-fuzzy theory is defined to be a mapping X: $\mathbf{F} \rightarrow L$ which maps each well-formed formula ϕ to an element in L, i.e., assigns a truth value $X\phi$ to ϕ. Unlike an interpretation, a theory X does not have to be a homomorphism. We write $X \leq Y$ if $X\phi \leq Y\phi$ for each formula ϕ. Given a set S of mappings from $\mathbf{F}$ to L, the *intersection* of S is defined to be the mapping $X = \cap S$, where $X\phi = \inf\{T\phi \mid T \in S\}$. The semantic closure of an L-theory X is the unique theory $\Delta(X) = \cap S$, where $S = \{T \mid T \in \Sigma \text{ and } T \geq X\}$ and Σ = the set of all homomorphisms.

One similarly defines the notion of a syntactic closure of a theory based on a given set of axioms. First, an L-syntax is defined to be a pair $\langle A, R\rangle$, where A is the set of axioms which is nothing but an L-theory and R is a set of fuzzy inferencing rules. A fuzzy inference rule r has the form $\langle r', r''\rangle$, where r' is a (partial) mapping from $\mathbf{F}^n$ to $\mathbf{F}$ and r'' is a mapping from L^n to L. Here, r' describes how to infer a formula from n given formulas and r'' describes how to compute the truth value of the result from those of the given formulas. The fuzzy Modus-ponens rule shown in Example 1 is described in this notion as $r' = \dfrac{\langle A, A \rightarrow B\rangle}{\langle B\rangle}$ and $r'' = \dfrac{\langle p, q\rangle}{\langle p \otimes q\rangle}$. For a given L-syntax $\sigma = \langle A, R\rangle$ and a given L-theory X, the syntactic closure $\sigma(X)$ is the unique smallest L-theory such that $\sigma(X) \geq X$, $\sigma(X) \geq A$, and $\sigma(X)$ is *closed* w.r.t. the rules in R. Here, a theory X is called closed w.r.t. a rule $r = \langle r', r''\rangle$ if for any formulas $\phi_1, \phi_2, ..., \phi_n$ such that $r'(\phi_1, ..., \phi_n) = \phi$, the truth value $X\phi \geq r''(X\phi_1, X\phi_2, ..., X\phi_n)$. The syntactic truth value $\sigma(X)\phi$ is the supreme of derived truth values of ϕ over all derivations. An L-syntax $\sigma = \langle A, R\rangle$ is called sound and complete if $\sigma(X)$ coincides with $\Delta(X)$ for every L-theory X.

The following example illustrates the notion of $\Delta(X)$.

Example 3. Let A and B be the only basic propositions in our language (universe), and let X be a theory that assigns truth values $X(A \rightarrow B) = 1$ and $X\phi = 0$ for all other formulas ϕ in A and B. Shown below are the interpretations $T \geq X$ and the resulting $\Delta(X)$:

X	0	0	0	0	1	0	0	...
	A	B	$\neg A$	$\neg B$	$A \rightarrow B$	$B \rightarrow A$	$\neg A \rightarrow B$	...
T	$0 \leq a \leq 1$	$a \leq b \leq 1$	$1 - a$	$1 - b$	1	$1 - b + a$	$\min\{1, a + b\}$	...
$\Delta(X)$	0	0	0	0	1	0	0	...

There are several important points to note. First, the semantic closure $\Delta(X)$ of X is not a homomorphic mapping from $\mathbf{F}$ to [0, 1], since it gives truth value 0 for for both A and $\neg A$. Second, if we consider $X = \{A \rightarrow B\}$ as a theory in ordinary boolean logic, then we would not define any truth value for A or B because there are models of $\{A \rightarrow B\}$ in which $A = 1$ (true) and models in which $A = 0$; the same is the case for B. However, in fuzzy logic, or precisely, in $\Delta(X)$ we do have truth values assigned to both A and B, namely, 0, and the same truth values for $\neg A$ and $\neg B$. Of course, as the definition of semantic closure indicates, the truth values 0 for A, $\neg A$, B, and $\neg B$ are only *lower-bounds* and they are not exact values. The bound 0 here simply means that nothing can be said about A and B in fuzzy theory X; this was also the case when we regarded X as a boolean theory. ♣

Example 4. Let X be the same theory as in Example 3, except that $XA = 0.6$, $X\neg A = 0.2$ and $X(\neg A \rightarrow B) = 1$. We also show below the interpretations $T \geq X$ and $\Delta(X)$:

X	0.6	0	0.2	0	1	0	1	...
	A	B	$\neg A$	$\neg B$	$A \rightarrow B$	$B \rightarrow A$	$\neg A \rightarrow B$	...
T	$0.6 \leq a \leq 0.8$	$a \leq b \leq 1$	$1 - a$	$1 - b$	1	$1 - b + a$	1	...
$\Delta(X)$	0.6	0.6	0.2	0	1	0.6	1	...

We have $\Delta(X)B = 0.6$, which is the semantic truth value of B from X. Syntactically, we can derive $\tau(B) \geq 0.6$ as follows. First, using the fuzzy modus ponens, from $\tau(A \to B) \geq 1$ and $\tau(A) \geq 0.6$ in X, we obtain $\tau(B) \geq \tau(A) \otimes \tau(A \to B) = 0.6$. We may have other derivations for the lower-bound of $\tau(B)$ (e.g., from $\tau(\neg \mathrm{A}) \geq 0.2$, $\tau(\neg A \to B) \geq 1$, we have $\tau(B) \geq 0.2$). The syntactic truth value $\sigma(X)B$ is taken to be the greatest lower-bound of $\tau(B)$ over all derivations, and we have $\sigma(X)B = 0.6$, the same as the semantic truth value of B. The syntactical truth values of other formulas are derived in the same way. ♣

The main result of Pavelka states that as far as the R-implications on [0,1] are concerned, only the fuzzy logics isomorphic to $L_{\aleph}$ possesses an L-syntax which is sound and complete w.r.t. such a logic.

Shown below is an isomorphic form of $L_{\aleph}$. We also show the usual definitions of "$\to$" and "$\neg$" in $L_{\aleph}$ side by side for ease of comparison; we also include the definitions for the related operators "$\vee$" and "$\oplus$". As we noted earlier "$\neg$" can be defined via "$\to$" since we have $A \to \bot \equiv \neg A$ (both for Boolean logic as well as for $L_{\aleph}$). In each case, we have $\tau(A \vee B) = \max\{a, b\}$, which is simply the usual lattice operation on the truth value set [0,1] with respect to the ordering "$\leq$". Recall that $A \vee B$ is an abbreviation for $(A \to B) \to B$ and $A \oplus B$ an abbreviation for $\neg A \to B$. For Boolean values, i.e., the truth-value set $\{0, 1\} \subset [0,1]$, we have $\tau(A \vee B) = \tau(A \oplus B)$; in general, we have $\tau(A \oplus B) \geq \tau(A \vee B)$, and hence the name "strong disjunction" for "$\oplus$".

Assume $\tau(A) = a$ and $\tau(B) = b$.

(1)	$\tau(A) = 1 - a$	$= \tau(A \to \bot)$
	$\tau(A \to B) = \min\{1, 1 - a + b\}$	$= \tau(\neg B \to \neg A)$
	$\tau(A \vee B) = \max\{a, b\}$	$= \tau((A \to B) \to B)$
	$\tau(A \oplus B) = \min\{1, a + b\}$	$= \tau(\neg A \to B)$
(2)	$\tau(\neg A) = \sqrt{1 - a^2}$	$= \tau(A \to \bot)$
	$\tau(A \to B) = \min\{1, \sqrt{1 - a^2 + b^2}\}$	$= \tau(\neg B \to \neg A)$
	$\tau(A \vee B) = \max\{a, b\}$	$= \tau((A \to B) \to B)$
	$\tau(A \oplus B) = \min\{1, \sqrt{a^2 + b^2}\}$	$= \tau(\neg A \to B)$

One defines $A \wedge B = \neg(\neg A \vee \neg B)$ and $A \otimes B = \neg(\neg A \oplus \neg B)$. Their truth-values are given by $\tau(A \wedge B) = \min\{a, b\}$ and $\tau(A \otimes B) = \max\{0, a + b - 1\}$. The operations $A \oplus B$ and $A \otimes B$ on the formulas suggest the following notations for the corresponding truth-values:

$$a \oplus b = \min\{1, a + b\} \text{ and } a \otimes b = \max\{0, a + b - 1\}.$$

In this case, the isomorphism connecting (1) to (2) is given by the one-to-one and onto mapping $\mathrm{f}(x) = x^2$ from [0,1] to [0,1] so that the new negation is $\mathrm{f}^{-1}(1 - \mathrm{f}(\mathrm{x}))$ and the new implication is $\mathrm{f}^{-1}(1 - \mathrm{f}(\mathrm{a}) + \mathrm{f}(\mathrm{b}))$.

A more general isomorphic form of $L_{\aleph}$ is obtained by using an arbitrary $p > 0$ in place of the exponent 2 and $1/p$ in place of the square-root operation. For instance, we can let $\tau(\neg A) = (1 - a^p)^{1/p}$ and $\tau(A \to B) = \min\{1, (1 - a^p + b^p)^{1/p}\}$. As $\mathrm{p} \uparrow \infty$, we have $\tau(A \oplus \mathrm{B}) = \min\{1, (a^p + b^p)^{1/p}\} \downarrow \max\{a, b\} = \tau(A \vee \mathrm{B})$.

It is a common error indeed [16] to use the specific negation operation $\tau(\neg A) = 1 - \tau(A)$ in combination with the more general definition of $\tau(A \oplus B) = \min\{1, (a^p + b^p)^{1/p}\}$. As we know now from Pavelka's work, this is meaningless and must not be done.

4.2 Defining the Truth-value $I(a, b)$

What is a good definition for the fuzzy implication operator? Pavelka's work [10] answered this question as far as R-implications over [0,1] interval are concerned. Here, we take another look at the fuzzy operator "$\to$" in terms of some of the tautologies that must hold in the resulting theory and use them to critique some of the alternative definitions proposed in the literature.

Let $I(a, b)$ denote an abstract definition of the truth-value $\tau(A\to B)$ given that $\tau(A) = a$ and $\tau(B) = b$. Table 2 shows two tautologies that should hold and the corresponding properties that $I(a, b)$ must satisfy.

Table 2. Two tautologies and the corresponding properties of $I(a, b)$.

	Tautologies	Properties of $I(a, b)$
(1)	$(A\to B)\to(\neg B\to\neg A)$	$I(a, b) = I(\neg b, \neg a)$
(2)	$(A\to(B\to C))\to(B\to(A\to C))$	$I(a, I(b, c)) = I(b, I(a, c))$

The first property (1) says that the formulas $A\to B$ and $\neg B\to\neg A$ are in fact equivalent since $A = \neg\neg A$ and $B = \neg\neg B$. Thus, $A\to B$ and $\neg B\to\neg A$ should have the same truth value, i.e., $I(a, b) = I(\neg b, \neg a)$. The second property (2) says that (by interchanging A and B) $(A\to(B\to C))$ is equivalent to $(B\to(A\to C))$. Thus, we should have $I(a, I(b, c)) = I(b, I(a, c))$.

It is easy to see that Lukasiewicz's implication function $I(a, b) = \min\{1, 1 - a + b\}$ satisfies both (1)-(2); $I(a, I(b, c)) = \min\{1, 2 - a - b + c\} = I(b, I(a, c))$. In [3], Dubois, Lang and Prade presented a detailed discussions about the properties one may desire of a fuzzy implication. It can be shown that not all R-implications satisfy both (1) and (2), and similarly not all S-implications satisfy both (1) and (2).

Shown below are some of the alternative definitions of $I(a, b)$ suggested in the literature (and justified by one intuitive consideration or the other) which fail to satisfy one or both of (1)-(2) or some other key property.

Table 3. Some improper choices for $I(a, b)$.

		Property (1)	Property (2)
Baldwin et al.[†]:	$I(a, b) = \min\{1, \dfrac{b(1-a)}{a(1-b)}\}$ (a variation of Gödel)	yes	no
Gödel:	$I(a, b) = \min\{1, \dfrac{b}{a}\}$	no $(a = 2/3, b = 1/2)$	yes
	in binary logic)		
Mamdani:	$\min\{a, b\}$	no	yes
Max-min rule:	$I(a, b) = \max\{1 - a, \min\{a, b\}\}$ (motivated by $A\to B = \neg A\vee(A\wedge B)$	no	no $(a = 1/3, b = 1/2, c = 2/3)$
Yager:	$I(a, b) = b^a$	no	yes
$S_\aleph$:	$I(a, b) = \begin{cases} 1, \text{ if } a \le b \\ 0, \textit{ otherwise} \end{cases}$	yes	no $(a = 1/2, b = 2/3, c = 1/3)$
$G_\aleph$:	$I(a, b) = \begin{cases} 1, \text{ if } a \le b \\ b, \textit{ otherwise} \end{cases}$	no	no $(a = 1/2, b = 2/3, c = 1/3)$

Zadeh-rule, $I(a, b) = \max\{1 - a, b\}$, which is motivated by $A\to B = \neg A\vee B$ in binary logic, satisfies both (1)-(2) but it has other problems, namely, for $a \le b$, $I(a, b)$ may not equal 1. Lukasiewicz's $I(a, b)$ does not suffer from this problem.

In [5], Lukasiewicz's rule $I(a, b) = \min\{1, 1 - a + b\}$ is abandoned incorrectly in favor of $S_\aleph$ and $G_\aleph$ (called "standard sequence" and "Gödelian sequence" [11]) due to a common misconception about the operators "$\wedge$" and "$\otimes$". The argument given in [5] is that $a\wedge I(a, b) = \min\{a, I(a, b)\} \nleq b$; here one should have used the operation $a\otimes b = \max\{0, a + b - 1\}$ associated with $I(a, b) = \min\{1, 1 - a + b\}$ in

[†] In [2], the authors suggest a more general form $I(a, b) = \min\{1, [b(1 - a)]^p/[a(1 - b)]^p\}$, $p > 0$, and justify them on the basis of several other intuitive properties while ignoring the property (2) which is not satisfied.

place of "∧", i.e., one should have verified that $a\otimes I(a, b) = \max\{0, a + \min\{1, 1-a+b\} - 1\} = \max\{0, \min\{a, b\}\} = \min\{a, b\} \leq b$ as desired. The use of S_K and G_K in combination with Zadeh's composition rule are given several intuitive justifications in [5] along the lines similar to those in [2] for $I(a, b) = \min\{1, [b(1-a)\}/[a(1-b)]\}$. But as we observed in Table 3 those justifications are inadequate because of its failure to satisfy property (2).

5. CONCLUSIONS

In this paper, we clarify some of the confusions about the Lukasiewicz logic and fuzzy logic. We point out the key differences between inferencing in the Lukasiewicz logic and in fuzzy logic and the need to define the strong conjunction ⊗ in connection with the fuzzy modus ponens inference rule. Key properties of a suitable fuzzy implication operator "→" are discussed and the importance of the Lukasiewicz implication is re-emphasized, together with a critique of some alternative definitions of fuzzy implication. This is in further support of Pavelka's results that for fuzzy logics on the unit interval [0,1] with R-implications, the Lukasiewicz implication and its isomorphic forms are the only suitable choice.

6. REFERENCES

[1] Ackermann, R., *An Introduction to many-valued logics*, Dover Pub. Inc., New York, 1967.

[2] Baldwin, J.F. and Pilsworth, B.W., Axiomatic approach to implication for approximate reasoning with fuzzy logic, *Fuzzy Sets and Systems*, 3(1980), pp. 191-219.

[3] Dubois, D., Lang, J., and Prade, H., Fuzzy sets in approximate reasoning, Part 1: inference with possibility distributions, *Fuzzy Sets and Systems*, 40(1991), pp. 141-202.

[4] Dubois, D., Lang, J., and Prade, H., Fuzzy sets in approximate reasoning, Part 2: logical Approaches, *Fuzzy Sets and Systems*, 40(1991), pp. 203-244.

[5] Fukumi, S., Mizumoto, M., and Tanaka, K., Some considerations of fuzzy conditional inference, *Fuzzy Sets and Systems*, 4(1980), pp. 243-273.

[6] Goguen, J.A., The logic of inexact concepts, *Synthese*, 19(1968), pp. 325-373.

[7] Gottwald, S., Fuzzy propositional logics, *Fuzzy Sets and Systems*, 3(1980), pp. 181-192.

[8] Lukasiewicz, J., *Selected works - Studies in logic and the foundations of mathematics*, North-Holland, Amsterdam, Warsaw, 1970.

[9] Mamdani, E.H., Application of fuzzy logic to approximate reasoning using linguistic systems, *IEEE Trans. Computers*, C-26 (1977), pp. 1182-1191.

[10] Pavelka, J., On fuzzy logic I, II, III, *Zeitsch. Math. Logik*, 25(1979), pp. 45-52, 119-134, 447-464.

[11] Rescher, N., *Many-valued Logic*, McGraw-Hill Inc. 1969.

[12] Smets P. and Margrez, P., Implication in fuzzy logic, *Intern. J. of Approximate Reasoning*, 1(1987), pp. 327-347.

[13] Trillas, E. and Valverde, L., On mode and implication in approximate reasoning, in *Approximate Reasoning in Expert Systems* (M.M. Gupta et al., eds.), Elsevier Sci. Pub., Amsterdam, 1985.

[14] Turunen, E., Algebraic structures in fuzzy logic, *Fuzzy Sets and Systems*, 52(1992), pp. 181-188.

[15] Wajsberg, M., Axiomatization of the 3-valued propositional calculus, (1931), Translated in S. McCall (ed.), *Polish Logic: 1920-1939*, Oxford, (1967), pp. 264-284.

[16] Yager, R.R., On a general class of fuzzy connectives, *Fuzzy Sets and Systems*, 4(1980), pp. 235-242.

[17] Zadeh, L., Fuzzy sets, *Information and Control*, 8(1965), pp. 338-353.

Logic Systems for Approximate Reasoning: via Rough Sets and Topology

T.Y. Lin[1], Qing Liu[2] and Y.Y. Yao[3]

[1] Department of Mathematics and Computer Science
San Jose State University, San Jose, California 95192
[2] Department of Computer Science
Stanford University, Stanford, CA 94305
(On leave from Jiangxi University
Nan-Chang, Jiangxi, P.R. China)
[3] Department of Mathematical Sciences, Lakehead University
Thunder Bay, Ontario, Canada P7B 5E1

Abstract. Mathematicians formalized the approximation in terms of topology. In this paper a new family of logic systems for approximate reasoning, called **Near Logic**, is proposed; their semantics are rested on the notion of neighborhood system–a building block of topology. Somewhat surprisingly, the axiom schema of the Near Logic is that of the modal logic S_4. This generalizes the fact that the axiom schema of Rough Logic is S_5. The agreement in geometric and modalic considerations seems indicate that the proposed approach must have captured some intrinsic meaning of the approximate reasoning. Neighborhood Systems are very general approximation, so Near Logic can be regarded as a small step toward the formalization what Hao Wang called "approximate proof" three decades ago.

1 Introduction

Approximation is a fact of life; daily routines are carried out by approximation. Scientific and engineering operations, such as numerical analysis, signal processing are all processed by approximation. Theoretical foundation of such approximations is well developed in the theory of topological spaces or more generally $Freshet(V)$ spaces [15]. Traditionally all computer systems are based on "exact" mathematics, where the fundamental principles are clear, and crispy. As computer systems have matured and applications have expanded, the requirements of handling the vagueness, imprecise become imperative. Around 1986, the first author began to introduce the notion of neighborhood systems (the building block of topology) into databases (approximate retrieval) and knowledge bases (approximate reasoning) [2], [3], [4], without realizing that the rough sets theory, has been developed extensively by Pawlak school. Rough set theory induces a special type of topology in its universe, called Pawlak topology [6],where open sets coincide with closed sets.

Topological spaces (neighborhood systems) can be characterized by Kuratowski's closure axioms [7]. Based on these characterization, a new family of formal logical systems, called **Near Logic (Rough Logic)** is proposed and studied. A detail report on the rough logic will soon be reported in [16]. Here, we focus on the Near Logic and its comparison with Rough Logic. Interestingly, the axioms schema of Near and Rough logic are that of modal logic S_4 and S_5 respectively. However, we should caution the readers that the respective models are quite different. Our approach seems to have captured some intrinsic nature of approximate reasoning, because two independent considerations, geometric and modalic, lead to the same axioms. Neighborhood system is a very general approach to the approximation problem, so this paper can be viewed as a small step toward a formalization of what Hao Wang called "approximate proof" [18].

2 Neighborhood Systems and Rough Sets

In modern mathematics, mathematician generalized the concept of "distance" into the "neighborhood systems", and create the theory of topological spaces. In his book, Sierpenski defined a space called $Frechet(V)$ space which is more general than topological space [17]. *$Frechet(V)$ space is a space in which every point has a neighborhood system with no further axioms.* In topological space, we do require some axioms on the neighborhood system. It should be pointed out here that the rough set theory also give us a special type of topology–called Pawlak topology [6]. The collection of equivalence classes forms the base of Pawlak topology.

Intuitively neighborhood systems handle the notion of "close to", "similar to", or "approximate to". Such notion in general may not be transitive, and hence can not be handled by rough set theory, which is based on equivalence relation. For example, East LA is "close to" LA, LA is "close to" West LA. However, East LA is not considered to be "close to" West LA. So when applications call for non-transitive approximation, we may need to use neighborhood systems (near logic), instead of rough set theory (rough logic).

Definitions
A **$Frechet(V)$ space** U is a set of points in which every point has associated with at least one subset of U, called neighborhood.
A **topological space** U is a $Frechet(V)$ space, where the family of neighborhoods satisfy topological axioms.
A **neighborhood system on** U is a set of points in which every point is associated with one (fixed) neighborhood.

The family of neighborhoods in $Frechet(V)$ space or topological space is called F-topology or topology respectively [17]. $Frechet(V)$ space is very general: Let p be a point in U. A neighborhood of a point p is a subset of U. In general, the subset may or may not contains p. However, it can be shown that F-topolgically, there is no loss in generality to assume that all neighborhoods do contain p [17].

A neighborhood system is a special type of $Frechet(V)$ space, but is not a topological space. A topological space is also a $Frechet(V)$ space, but not vice versa. There can have many neighborhood system in one $Frechet(V)$ space, or one topological space. For the motivation of introducing neighborhood systems, let us quote from [4] that a neighborhood system is "....an instance of topology, not a genuine topological space. It has neighborhoods of tolerance for all its entities. In numerical analysis, before the approximation begins, a radius of tolerance (i.e., neighborhoods of tolerance) has to be chosen. And during the process of finding an approximate solution, the only relevant neighborhood is the chosen one." we beleive neighborhood system is a correct setting for such approximation.

Surprisingly, the notion of neighborhood systems is very closely related to general modal logic system, we will investigate in the future papers. In this paper, we will focus on the neighborhood systems which are instances of topology. If a topology is defined by a partition (equivalence classes), the neighborhood system is called the **neighborhood system of rough sets or simply the rough sets**, by abuse of language.

Intuitively, the first-order classical logic is the logic system for set theory. We are trying to build a logic system for the topological spaces. So we would like to "characterize" the category of topological spaces, so that one can describe the category via formulas. It is well-known that Kuratowski closure axioms characterize the topology, [1],[15]. The approximate operators can be defined by

$L(X) = interiorof\ X = \{p$:N(p)is a subset ofX}
$H(X) = closureof\ X = \{p$:N(p)andXhas nonempty intersection }.

It is clear that a continuous inclusion (the inclusion in the category of topological spaces) induces an inclusion on its interior points (the reverse may not be true), so we propose axiom (7). The Low Inclusion, denoted by $\subseteq_{(low)}$, is the induced inclusion on the interior points.

(7.) Low inclusion: $X \subseteq_{(low)} Y$ implies $L(X) \subseteq L(Y)$.

In axiomatic set theory, an arbitrary set X corresponds to a 'property', a well-formed formula (wff). The correspondence can be expressed by $X = \{x \mid w(x)\}$. So the Kuratoeski axiom can be rewritten as formulas:

(1"). $\Xi = false \leftrightarrow H\Xi = false$, where Ξ is a empty formula;
(2"). $Lw \rightarrow w$;
(3"). $H(w_1 \vee w_2) \leftrightarrow Hw_1 \vee Hw_2$;
(4"). $Lw \rightarrow LLw$;
(5"). $\sim Lw \leftrightarrow H \sim w$;
(7"). $L(w_1 \rightarrow w_2) \rightarrow (L(w_1) \rightarrow L(w_2))$.

These well formed formulas are six axioms for the category of topological spaces. They will be adopted as the axiom sxheme of a new logical system, called Near Logic. In rough set theory, the approximate operators induce a Pawlak topology on U, where open sets are also closed sets. This fact, expressed in the following formula, is included into the axiom scheme of rough logic

(6"). $Lw \leftrightarrow HLw$.

3 The Near and Rough Logic

As usual, we need symbols: P is a set of predicate symbols, CO is a set of constant symbols, FU is a set of partial functions symbols, X is a set of individual variables, RO is a set of (modal) approximate operators. The term and formula can be defined in usual way. Let Ω be the language of rough and near logic.

Axiom Systems
1. The language Ω
2. Axiom schemes:

Axioms for Near Logic

A_1-A_5: The axioms of first order logic and

A_6: $\vdash L(w_1 \rightarrow w_2) \rightarrow (Lw_1 \rightarrow Lw_2)$;

A_7: $\vdash Lw \rightarrow w$;

A_8: $\vdash Lw \rightarrow LLw$;

The numbering scheme of these axioms is synchronized with [16]

Axioms for Rough Logic

A_1-A_8 and

A_9: $\vdash Hw \rightarrow LHw$.

3. Rule of Inferences

R_1: Modus Ponens (MP): From $\vdash w_1 \rightarrow w_2$ and $\vdash w_1$, we get $\vdash w_2$;

R_2: L Insertion (LI): From $\vdash w$, we get $\vdash Lw$;

R_3: Generalization: From $\vdash w$, we get $\vdash (\forall x).w$.

It is clear that these axioms agree with the axioms of S_4, and S_5 [10] [8]:

Theorem. The Near and Rough Logic is equivalent to the modal logic S_4 and S_5 respectively.

4 The Worlds of Near Logic

We adopt the relational structure from the classical first-order logic [11]; however, we consider partial functions instead of full functions. Moreover, there is a topology in the universe U defined by the approximate operators. For each entity e in U, one can choose a fixed neighborhood $N(e)$. In other words, *a neighborhood system, or an instance of topology, is chosen.* A **near relation** R can be defined for a chosen neighborhood system as follows:

y is **near** x, denoted by $x \ \ R \ \ y$, iff y is in the chosen neighborhood $N(x)$ of x. It should point out here that R is not necessary reflexive, symmetric or transitive; these properties depend on how the $N(x)$ is chosen.

4.1 Observable Worlds

The axioms of L and H define a topology on U. Let B be a base for the topology so defined. Recall that a base is a collection of open sets, call base open sets, such that, any open set in U is a union of the base open sets [1]. For every entity $e \in U$, we can choose a base open set as a chosen neighborhood $N(e)$ of the entity e. It is clear that **not** all N(e) are distinct, let

$$OpenCov{=}N(e_i),\ i = 1, 2, \ldots$$

be the set of all the distinct members of $N(e)'s$, where each $N(e)$ is a base open set. Let

$H(e) = H(N(e))$ denotes the "closure" of $N(e)$,

where H is the upper approximate operator. It is clear that not all $H(e)$, $e \in U$, are distinct, let

$$Cov{=}H(e_i),\ i = 1, 2, \ldots$$

be the set of all the distinct members of $H(e)'s$. Such Cov is a (chosen) neighborhood system of U; it is a covering of U i.e., $U = \bigcup H(e_i)$, $i = 1, 2, \ldots$.

(1) The Case for Rough Sets: If H and L satisfy the "six axioms" of rough sets [7], the collection $Cov{=}H(e_i) = H_i$, $i = 1, 2, \ldots$ is a partition on U. Cov is the neighborhood system of rough sets. Let W^h, $h > 0$ be a set of representative, that is, it consists of one and only one representative from each H_i. Let W, called the set of observable worlds, be the collection of all such representative W^h. Each W^h, $h > 0$ has an induced relational structure from U.

(2) The Case for Neighborhood Systems: The collection

$Cov{=}H(e_i)$, i=1,2,... is a neighborhood system on U.

For the chosen Cov, let $W^h(Cov)$, $h > 0$ be a set of representative, that is, it consists of one and only one representative from each $H(e_i)$. We will use $W(Cov)$ to denote the collection of all such representative $W^h(Cov)$, $h > 0$, and W to denote the total collection for all neighborhood systems, namely,

$$W = \bigcup W(Cov) \ \ \forall Cov.$$

Intuitively, a neighborhood system Cov (or a partition) represents one knowledge representation or one experiment, and each $W^h(Cov)$ represent one particular observation of such experiment. As in the previous case, we will give each $W^h(Cov)$ an induced relational structure.

In rough set theory, there is one "minimal" neighborhood system, so it is adequate to study one neighborhood system (namely, the partition). In general, one neighborhood system of a topological space is not adequate for approximation problem, we have to consider all neighborhood systems (constructed from the topological space U); note that the axioms on L and H characterize the topology, but not any particular chosen neighborhood system.

4.2 Nearly (or Roughly) Distinguished Entities

In rough set theory, an entity which is indiscernible from a distinguished entity, is regarded as distinguished in that observable world (but may not be in the actual world); they are called roughly distinguished. In a neighborhood system, an entity which is near the distinguished entity (i.e., it is belong to the neighborhood of the distinguished entity), should also be regarded as distinguished in that observable world. They will be called **nearly distinguished entities**.

4.3 Accessibility Relations

A binary relation AR on the set of observable worlds W is called an **"accessibility relation"** iff for any W^i, $W^j \in W$, such that $< W^i, W^j >\in AR$ and W^j is possible with respect to W^i [11].

From the construction of the observable worlds W^h, $h = 1, 2, \ldots$, we will be able to show that an accessibility relation exists in the set W of observable worlds. Each observable world W^h, $h = 1, 2, \ldots$ consists of one representative from each $H_i(e)$ (a neighborhood of e). So we define an accessibility relation AR as follows:

(1)The case for Rough Sets:
Let W^h, W^k be any two observable worlds. $< W^h, W^k >\in AR$ iff for every $x \in W^h$, and every H_i there is a unique $y \in W^k$ such that both x and y are in $H_i(e)$ (or equivalently $< x, y >\in R$). It is easy to see that AR is an equivalence relation [16]

(2)The case for Neighborhood Systems:
(2.1) Accessibility Relation in a Neighborhood System Cov:
Let $Cov = H(e_i) = H(N(e_i))$, i=1,2, ... be a neighborhood system of U. Let $W^h(Cov)$, $W^k(Cov)$ be any two observable worlds. The pair of observable worlds $< W^h(Cov), W^k(Cov) > \in AR$ iff for every $x \in W^h(Cov)$, and every $H_i \in Cov$ there is a $y \in W^k$, in other words, there is a function $f : W^h(Cov)$: $\rightarrow W^k(Cov)$, which maps x to y

It is easy to see that AR is a reflexive (note e is in $N(e)$, when U is a topological space). However, it may not be symmetric: If y is in the intersection of two $H(e_i)$'s, then the inverse f^{-1} is not a function. Since function is transitive relation, so AR is transitive.

(2.2) Accessibility Relation in General: Let Cov_1 be a refinement of Cov_2, The we can define a function $F_{Cov} : Cov_1 \longrightarrow Cov_2$ as follows: $F_{Cov}(N_1) = N_2$, if $N_1 \in Cov_1$ is a subset of $N_2 \in Cov_2$. The function F_{Cov} induces a function f: $W^h(Cov_1) \longrightarrow W^k(Cov_2)$ as follows: $f(x) = y$, if the neighborhood of y (i.e., y is the representative of that neighborhood in Cov_2) contains the neighborhood of x in Cov_1 It is easy to see that AR is reflexive and transitive.

5 Models for Near and Rough Logic

In this section we are going to discuss its semantics. Our method is essentially similar to the classical modal theory. However, there are some deviation on the assignment α, and interpretations γ. Note that α may assign a variable to a value in U, which may not exit in the given W^h. So some assigned values may have to be replaced by a near one(replaced by an element in its neighborhood). From the construction of W^h, there alway exits a near one in the W^h, so the induced assignment α^h will assign the variable to that near value. In the background, the near model (or rough model) is induced from the classical first-order model. However, the user can only see the observable worlds, the model for modal logic. As the accuracy of the observations increases, we may get the first order theory back–this is the convergency theory; its report is defered to later paper.

5.1 Near (or Rough) Structures

A Near (or Rough) Structure is a 3-tuple

$$W = (W, U, RO)$$

where:

U is a relational structure, called **ideal world.** We may write $U = W^\infty$ to emphasize that U is unobservable and is the final "limit" of the approximation.
RO is a set of approximate operators (near or rough), which satisfy the five axioms (or six axioms) on U, i.e., $RO = H$;
W is a collection of observable worlds, each of which has a relational "substructure" of U defined as follows:

$$W^h = (W^h, N^h, R^h, F^h);$$

where

(1). W^h consists of one representative from each class $H_i, i = 1, 2, \ldots$, called **Observable World** ;
(2). $U = \bigcup W^h$; it is the set of all entities;
(3). $N^h = H(N) \cap W^h$; $H(N) = \bigcup H(c)\ \ \forall c \in N$ is the upper approximation

of N. An entity which is near (or equivalent to) a distinguished entity (in ideal world) should be a distinguished entity in that observable world; they are called nearly (or roughly) distinguished entities.

(4). R^h consists of all the relations in R whose domains are restricted to W^h;

(5). F^h consists of all the partial functions whose domains are restricted to W^h.

(6). $W^0 = (U, H_i, i = 1, 2, \ldots)$, called **Approximation World**

The relational structure on W^0 is the "union" of that of W^h, $h > 0$; it is not "complete". Intuitively, the functions and predicates in W^0 are glued together by individual "component". We will skip the details of "union"; its relational structure is incomplete or "over" supplied by observations.

5.2 Near and Rough Models

A Near/Rough Model is defined as

$$M = (W, \gamma, \alpha),$$

where:

(1)W is a set of observable worlds W^h, $h = 1, 2, \ldots$;

(2)γ is an interpretation. In order to express the validity of formulas, we define an the interpretation from the language to the models. On each observable world W^h, we assign to each n-ary predicate in the language Ω a n-ary relation in R^h; to each function symbol in the language Ω, we assign a n-ary function in F^h, and to each constant in Ω a distinguished entity in N^h. The function γ which defines these assignments is called an interpretation to W^h.

(3) α is a valuation. A valuation is a function, which assigns each variable of the associated language Ω an entity of W^h.

The semantic value of an expression *exp* with respect to observable world W^h, an interpretation γ, and a valuation α, is denoted by

$$[[exp \mid W^h, \gamma, \alpha]].$$

The satisfaction of a formula w of language Ω by an observable world W^h, an interpretation γ, and a valuation α in $M = (W, \gamma, \alpha)$ is denoted by:

$$M \models w[h, \gamma, \alpha].$$

The usual notations will be used without remakrs. For lower and approximate operators we note that

1. $M \models Lw[h, \gamma, \alpha]$ iff for every k, if W^k is accessible with respect to W^h in the sense of rough sets, then $M \models w[k, \gamma, \alpha]$;

2. $M \models Hw[h, \gamma, \alpha]$ iff there are some k, $k = 1, 2, \ldots$, W^k is accessible with respect to W^h, and such that $M \models w[k, \gamma, \alpha]$.

Here, h and k are the indices of observable worlds. A formula w is said to be valid if it is true in every observable world of the model. Having defined valid formulas, we naturally look for a approximate deductive system.

6 Approximate Reasonig Systems

We will write $L \subseteq (X, Y)$ and $L(w_1 \rightarrow w_2)$ as $X \Rightarrow Y$ and $w_1 \Rightarrow w_2$ respectively, where $\Rightarrow$ is thought of as the abbreviation of L and $\subseteq$ or $\rightarrow$.

The Near Logic Approximate Reasoning System $NLARS$
Its axioms include
(1). Axioms from first-order Language F;
(2). Logical Axioms:
A_6: $\vdash (w_1 \Rightarrow w_2) \rightarrow (Lw_1 \rightarrow Lw_2)$;
A_7: $\vdash Lw \rightarrow w$;
A_8: $\vdash w \rightarrow LHw$.
(3). Rule of Inferences
NR_1: Near Modus Ponens (NMP): If $\vdash w_1 \Rightarrow w_2$ and $\vdash Lw_1$, then $\vdash Lw_2$;
NR_2: L Insertion (LI): If $\vdash w$ then $\vdash Lw$;
NR_3: If $\vdash w$ then $(\forall x)w$.
Sample Theorems in $NLARS$:
(T_1). $\vdash Lw \Rightarrow w$;
(T_2). $\vdash w \Rightarrow Hw$;
(T_3). $\vdash Lw \Rightarrow Hw$;
(T_4). $\vdash L(w_1 \wedge w_2) \Leftrightarrow Lw_1 \wedge Lw_2$;
(T_5). $\vdash H(w_1 \vee w_2) \Leftrightarrow Hw_1 \vee Hw_2$;
(T_6). $\vdash Lw_1 \vee Lw_2 \Rightarrow L(w_1 \vee w_2)$;
(T_7). $\vdash H(w_1 \wedge w_2) \Rightarrow Hw_1 \wedge Hw_2$;
(T_8). $\vdash (w_1 \Rightarrow w_2) \wedge w_1 \Rightarrow w_2$;
(T_9). $\vdash (w_2 \wedge w_1 \Rightarrow w_3) \Rightarrow (w_2 \Rightarrow (w_1 \rightarrow w_3)$.

7 Conclusions

The axioms of rough sets and neighborhood systems led us to define Near and Rough Logics. The two logic systems allow us to establish the "nearly correct" or "roughly correct" deducting systems. Using these systems, we can conclude some theorems which are approximately correct in the ideal world. We will explore some applications in the near future.

References

1. J. Kelley, General Topology, van Norstrand, 1955.
2. T.Y. Lin and S. Bairamian, Neighborhood systems and Goal Queries, Manuscript, California State University, Northridge, 1987.

3. T. Y. Lin, Neighborhood Systems and Relational Database. Proceedings of CSC '88, February, 1988.
4. T. Y. Lin, Neighborhood Systems and Approximation in Database and Knowledge Base Systems, Proceedings of the Fourth International Symposium on Methodologies of Intelligent Systems, Poster Session, October 1215, 1989.
5. T. Y. Lin, Q. Liu, K. J. Huang and W. Chen. Rough Sets, Neighborhood Systems and Approximation, Fifth International Symposium on Methodologies of Intelligent Systems, Selected Papers, Oct. 1990.
6. T.Y. Lin, A topological and Fuzzy Rough Sets, Intelligent Decision Support, Kluwer Academic Publishers, Boston, 1992, Chapter 5, Part II. edited by Slowinski.
7. T.Y. Lin and Q. Liu, Rough Approximate Operators, Axiomatic Rough Set Theory, Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, RSKD' 93, First Edition, Banff, Canada, 1993.
8. W. Lukaszewicz, Non-monotonic Reasoning–Formalization of Common Sense Reasoning, Institute of Information, University of Poland, 1990.
9. Z. Pawlak, Rough Sets, Theoretical aspects of reasoning about data, Kluwer Academic Pub., Dordrecht, 1991.
10. Brian F, Chellas, Modal Logic: an Introduction, Published by the Press Syndicate of the University of Cambridge, 1980.
11. Richad Frost, Introduction to Knowledge Base Systems, Macmillab Publishing Company, New York, 1986.
12. Akira Nakamura, On a Logic of Information for reasoning about Knowledge, Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, RSKD"93, First Edition, Banff, Canada, 1993.
13. Ewa Orlowska, Rough set semantics For Nonclassical Logics, Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, RSKD' 93, First Edition, Banff, Canada, 1993.
14. Helena Rasiowa and Andrzej Skowron, Rough Concepts Logic, Institute of Mathematics Warsaw University Rkin IX/907, 00-901 Warsaw Poland.
15. Sierpenski, General Topology, University of Torranto Press, 1960.
16. T. Y. Lin and Q. Liu, First Order Rough Logic I: Approximate Reasoning via Rough Sets, Fundamenta Informaticaie, 1994.
17. Sirepenski, General Topology, University of Torranto Press, 1956.
18. Wang, Hao: Toward Mechanical Mathematics, IBM Journal, 1960

Signed Formulas and Fuzzy Operator Logics

James J. Lu[1], Neil V. Murray[2], and Erik Rosenthal[3]

[1] Bucknell University, Lewisburg, PA 17838
[2] State University of New York, Albany, NY 12222
[3] University of New Haven, West Haven, CT 06516

Abstract. The language of signed formulas offers a classical logic framework for multiple-valued logics. Fuzzy operator logic is cast in this framework, thereby making most classical inference techniques applicable to fuzzy logic. In particular, resolution for first order fuzzy operator logic is shown to be a special case of signed resolution.

1 Introduction

A variety of non-standard logics (see, for example [2, 3, 5, 6, 8, 9, 15, 16]), have been studied in recent years. One feature common to much of this work, including our own, is the use of *signs*. The notion of a set of truth values as a sign is present in both Suchón's [17] and Doherty's [3] work, but only implicitly. The explicit and formal development of sets as signs was introduced independently in [5] by Hähnle and in [14]. This approach allows the utilization of classical techniques for the analysis of non-standard logics. This is appealing since, at the meta-level, human reasoning is essentially classical. That is, regardless of the domain of truth values associated with a logic, at the meta-level humans interpret statements about the logic to be either *true* or *false*.

In [12] the language of signed formulas was shown to have sufficient expressive power to encompass annotated logics [2, 8, 9]. In this paper, we cast fuzzy operator logic (FOpL) as defined in [19, 11] in the framework of signed formulas, thereby making several classical inference techniques — resolution, path dissolution and the tableau method, for example — applicable to FOpL's. In Section 3 fuzzy operator logic are defined as multiple-valued logics, and the fuzzy resolution rule developed in [19, 11] is shown to be a special case of signed resolution. (A good general introduction to fuzzy logic can be found in [1, 4].) In the next section, the language of signed formulas is reviewed, and in Section 2.2, the manner in which classical inference techniques can be adapted to signed formulas is described.

2 Multiple-Valued Logics and Signed Formulas

We assume a language Λ consisting of logical formulas built in the usual way from a set $\mathcal{A}$ of atoms, a set of connectives, and a set of logical constants. Associated with Λ is a set Δ of truth values, and an interpretation for Λ is a function from $\mathcal{A}$ to Δ; i.e., an assignment of a truth value to each atom in $\mathcal{A}$. A connective Θ

of arity n denotes a function $\Theta : \Delta^n \longrightarrow \Delta$. Interpretations are extended in the usual way to mappings from formulas to Δ. Alternatively, a formula $\mathcal{F}$ of Λ can be regarded as denoting a mapping from interpretations to Δ.

We use the term ***sign*** for any subset of Δ. A ***signed formula*** is defined to be an expression of the form $S{:}\mathcal{F}$, where S is a sign and $\mathcal{F}$ is a formula in Λ; if $\mathcal{F}$ is an atom in Λ, we call $S{:}\mathcal{F}$ a ***signed atom***. We are interested in signed formulas because they represent queries of the form, "Are there interpretations under which $\mathcal{F}$ evaluates to a truth value in S?" In a refutational theorem proving setting for classical logic, the query is typically $\{true\}{:}\mathcal{F}$, where the formula $\mathcal{F}$ is the negation of a goal or conclusion, conjoined with some axioms and hypotheses; the answer hoped for is no. To answer arbitrary queries, we map formulas in Λ to formulas in a classical propositional logic Λ_S, called the ***language of signed formulas***; it is defined as follows: The atoms are signed formulas and the connectives are (classical) conjunction and disjunction. We emphasize that a signed formula $S{:}\mathcal{F}$ is an atom in Λ_S regardless of the size or complexity of $\mathcal{F}$ and thus has no component parts in the language Λ_S. The set of truth values is of course $\{true, false\}$.

2.1 Λ-Consistent Interpretations

An arbitrary interpretation for Λ_S may make an assignment of *true* or *false* to any signed formula (i.e., to any atom) in the usual way. Our goal is to focus attention on those interpretations that relate to the sign in a signed formula. To accomplish this we restrict attention to ***Λ-consistent interpretations***. An interpretation I over Λ assigns to each atom, and therefore to each formula $\mathcal{F}$, a truth value in Δ, and the corresponding Λ-consistent interpretation I_{Λ_S} is defined by

$$I_{\Lambda_S}(S{:}\mathcal{F}) = \begin{cases} true & \text{if } I(\mathcal{F}) \in S \\ false & \text{if } I(\mathcal{F}) \notin S \end{cases} .$$

Note that this is a 1–1 correspondence between the set of all interpretations over Λ and the set of Λ-consistent interpretations over Λ_S. Intuitively, a Λ-consistent interpretation is one that assigns *true* to all signed formulas whose signs are simultaneously achievable via the corresponding interpretation over the original language. Restricting attention to Λ-consistent interpretations yields a new consequence relation: If $\mathcal{F}_1$ and $\mathcal{F}_2$ are formulas in Λ_S, we write $\mathcal{F}_1 \models_\Lambda \mathcal{F}_2$ if whenever I_{Λ_S} is a Λ-consistent interpretation and $I_{\Lambda_S}(\mathcal{F}_1) = true$, then $I_{\Lambda_S}(\mathcal{F}_2) = true$. Two formulas $\mathcal{F}_1$ and $\mathcal{F}_2$ in Λ_S are ***Λ-equivalent*** if $I_{\Lambda_S}(\mathcal{F}_1) = I_{\Lambda_S}(\mathcal{F}_2)$ for any Λ-consistent interpretation I_{Λ_S}; we write $\mathcal{F}_1 \equiv_\Lambda \mathcal{F}_2$. The following lemma is immediate.

Lemma 1. Let I_{Λ_S} be a Λ-consistent interpretation, let A be an atom and $\mathcal{F}$ a formula in Λ, and let S_1 and S_2 be signs. Then:
1. $I_{\Lambda_S}(\emptyset{:}\mathcal{F}) = false$;
2. $I_{\Lambda_S}(\Delta{:}\mathcal{F}) = true$;
3. $S_1 \subseteq S_2$ if and only if $S_1{:}\mathcal{F} \models_\Lambda S_2{:}\mathcal{F}$ for all formulas $\mathcal{F}$;
4. There is exactly one $\delta \in \Delta$ such that $I_{\Lambda_S}(\{\delta\}{:}A) = true$. □

Many classical inference rules begin with links (complementary pairs of literals). Such rules typically deal only with formulas in which all negations are at the atomic level. Similarly, the inference techniques used in this paper require that signs be at the "atomic level." To that end, we call a formula *Λ-atomic* if it has the property that whenever $S{:}A$ is an atom in the formula, then A is an atom in Λ; i.e., if the only atoms in the formula are signed atoms. Theorem 8 in Section 3.3 demonstrates that signs in fuzzy formulas can be driven to the atomic level in the cases in which we are most interested.

2.2 Signed Resolution

The key to adapting classical resolution to Λ_S is the next lemma, which follows immediately from part 4 of Lemma 1.

Lemma 2 (*The Reduction Lemma*). Let $S_1{:}A$ and $S_2{:}A$ be Λ-atomic atoms in Λ_S; then $S_1 : A \wedge S_2 : A \equiv_\Lambda (S_1 \cap S_2) : A$ and $S_1 : A \vee S_2 : A \equiv_\Lambda (S_1 \cup S_2) : A$. □

Consider a Λ-atomic formula $\mathcal{F}$ in Λ_S that is in conjunctive normal form (CNF). Let $C_j, 1 \leq j \leq r$, be clauses in $\mathcal{F}$ that contain, respectively, Λ-atomic atoms $\{S_j : A\}$. Thus each C_j has the form $K_j \vee \{S_j : A\}$. Then the *resolvent* R of the C_j's is defined to be the clause

$$\left(\bigvee_{j=1}^{r} K_j\right) \vee \left(\left(\bigcap_{j=1}^{r} S_j\right) : A\right) .$$

The rightmost disjunct is called the *remnant* of the resolution; it is called *impossible* if its sign is empty and *non-trivial* if it is not. We must augment this definition with the following obvious simplification rules that stem from the Reduction Lemma. First, if the remnant is impossible, it may simply be deleted from R. Secondly, whenever $S_1{:}B$ and $S_2{:}B$ are in R, we may *merge* them into the literal $(S_1 \cup S_2){:}B$; if $(S_1 \cup S_2) = \Delta$, then R is a tautology and may be deleted from $\mathcal{F}$. It is straightforward to prove that signed resolution is sound (see [15] for the details).

The classical notion of subsumption also generalizes to Λ_S: Clause C subsumes clause D if for every literal $S{:}A \in C$, there is a literal $S'{:}A \in D$ such that $S \subset S'$.

The appropriate versions of Herbrand's Theorem and of completeness take a bit of work because the set of truth values may be infinite, but they are essentially straightforward.

Theorem 3. Let $\mathcal{F}$ be a formula in Λ, and let S be a subset of Δ such that no interpretation maps $\mathcal{F}$ into S. Then there is a derivation of the empty clause from any Λ-atomic equivalent of $S{:}\mathcal{F}$ using signed resolution. □

3 Signed Formulas and Fuzzy Operator Logic

Fuzzy logic has been studied extensively, and several important applications have been developed, but, on the whole, relatively little has been done to explore practical inference methods for such logics. However, several authors ([10, 13, 11, 19], for example) have adapted resolution to fuzzy logic. Here we follow the treatment in [19, 11] and describe how the fuzzy resolution rule developed in those papers may be obtained from signed resolution. But the results imply much more: The logic of signed formulas is a classical logic in which interpretations are restricted to the Λ-consistent ones. While this restriction does mean that classical inference techniques do not automatically apply, we know of no inference rule that is not easily adaptable. Thus, the results of this section enable the application of a large collection of inference techniques to fuzzy logic.

3.1 Fuzzy Operators

In [19] a calculus called *fuzzy operator logic* was developed. It is a multiple-valued logic whose truth domain is the unit interval [0, 1]. The authors introduce a generalized negation that they call the *fuzzy operator*, and they use two binary connectives, $\wedge$ and $\vee$, to build their formulas. Formally, a *fuzzy formula* may be defined as follows. If A is an atom, if $\mathcal{F}$ and $\mathcal{G}$ are fuzzy formulas, and if $\phi \in [0,1]$[4], then

1. ϕA is a fuzzy formula;
2. $\phi(\mathcal{F} \wedge \mathcal{G})$ is a fuzzy formula;
3. $\phi(\mathcal{F} \vee \mathcal{G})$ is a fuzzy formula.

A simple example of a fuzzy formula is: $\mathcal{F} = A \wedge .3(.9B \vee .2C)$.

Several observations are in order. First, fuzzy operators are represented by real numbers in the unit interval. (That there are uncountably many fuzzy operators should not cause alarm. In practice, considering only rational fuzzy operators is not likely to be a problem. Indeed, with a computer implementation, we are restricted to a finite set of terminating decimals of at most n digits for some not very large n.) In particular, real numbers in the unit interval denote both truth values and fuzzy operators. Secondly, every formula and subformula has a fuzzy operator in front of it; any subformula that does not have an explicit fuzzy operator in front of it is understood to have 1 as its fuzzy operator.

A logical formula has no meaning until a semantics is attached; that is, a domain of truth values Δ must be selected, and n-ary connectives must be defined as functions from Δ^n to Δ. In an MVL in which Δ is a complete lattice, the most natural generalizations of the connectives $\wedge$ and $\vee$ are as the infimum and supremum operators. Should the truth domain be linear — for example, should it be $[0, 1]$ — then the infimum and supremum operators on finite sets represent the minimum and maximum. That is, the infimum and supremum of a finite set

[4] In [19] the authors use λ for the fuzzy operator and Λ for the threshold of acceptability (introduced in Section 3.2). We use ϕ and Γ to avoid confusion with notation used earlier in this paper and in previous papers.

in a linear domain are simply the minimum and the maximum, respectively. In [19] these connectives are not defined explicitly, but the definition of *confidence* (described in Section 3.2) amounts to defining $\wedge$ and $\vee$ as the infimum and supremum.

To define the fuzzy operator, a kind of "fuzzy" product is introduced in [19, 11][5]: If $\phi, \delta \in [0,1]$, then $\phi \otimes \delta = (2\phi - 1) \cdot \delta - \phi + 1$. Observe that $\otimes$ is commutative and associative. Also observe[6] that $\phi \otimes \delta = \phi \cdot \delta + (1-\phi) \cdot (1-\delta)$.

This last observation gives us the intuition behind the fuzzy product $\otimes$: Were ϕ the probability that A_1 is *true* and were δ the probability that A_2 is *true*, then $\phi \otimes \delta$ would be the probability that A_1 and A_2 are both *true* or both *false*. (This probabilistic analogy is for intuition only; fuzzy logic is not based on probability, but rather on possibility.)

The fuzzy product is used to define the fuzzy operators, which, we emphasize, are unary connectives that correspond to real numbers in the unit interval. For each $\phi \in [0,1]$, the fuzzy operator ϕ may be applied to any formula $\mathcal{F}$, producing $\phi\mathcal{F}$. To give a semantics to $\phi\mathcal{F}$, ϕ must be defined as a function from [0, 1] to [0, 1]. It will be convenient to do so using functional notation, so define Φ_ϕ by $\Phi_\phi(\delta) = \phi \otimes \delta$. Then, if δ is the truth value of $\mathcal{F}$ under some interpretation I (so that $\delta \in [0,1]$), the truth value of $\phi\mathcal{F}$ under I is $\Phi_\phi(\delta)$.

A good way to think of the fuzzy operator is as a confidence of the accuracy of the truth value of the formula. Thus, $1\mathcal{F}$ asserts the certainty of the assigned truth value of $\mathcal{F}$, $.8\mathcal{F}$ asserts that the assigned truth value is likely to be correct, and $.2\mathcal{F}$ asserts that the assigned truth value of $\mathcal{F}$ is likely to be wrong. It is important to be clear on the distinction between the fuzzy operator as a confidence of the accuracy of the truth value of a formula and the actual truth value of the formula. For example, if $\mathcal{F} = 0$ (*false*), then the truth value of $1\mathcal{F}$ is 0; i.e., we are certain that $\mathcal{F}$ is *false*. The truth value of $\phi\mathcal{F}$ may thus be thought of as the degree of confidence that $\mathcal{F}$ is *true*.

There are three fuzzy operators that play a special role: 0, .5, and 1. Let $\delta' = \phi \otimes \delta$. If $\phi = 1$, then $\delta' = \delta$, which in effect says that if we're certain of the truth value of $\mathcal{F}$, then $\mathcal{F}$ in fact has the truth value of $\mathcal{F}$ (not very deep). If $\phi = 0$, then $\delta' = 1 - \delta$. The 0 asserts certainty that the assigned truth value is wrong and converts the truth value to its complement. Another way to view this is that the fuzzy operator 0 is essentially standard negation. If $\phi = .5$, then $\delta' = .5$. This is not surprising: $\phi = .5$ means that we are completely uncertain of the truth value of $\mathcal{F}$.

We can better understand the semantics of fuzzy operator logic by distinguishing between a fuzzy operator in [0, .5) and a fuzzy operator in (.5, 1]. Let $\phi = .5 + \phi'$. Thus, $\phi' < 0$ in the former case, and $\phi' > 0$ in the latter. If δ is any truth value ($\delta \in [0,1]$), then manipulating $(\phi' + .5) \otimes \delta$ produces $\phi \otimes \delta = .5 - \phi' + 2\phi'\delta$. Observe first that $\phi \otimes \delta = .5$ iff $\phi' = 0$ or $\delta = .5$; i.e., iff $\phi = .5$ or $\delta = .5$. Also, as a function of δ, the fuzzy operator ϕ is monoton-

[5] Their choice of the operator $\otimes$ was well thought out: It produces exactly the right unary operators. They do not use the term *fuzzy product.*

[6] We would like to thank Don Fridshal for pointing out this relationship.

ically increasing or decreasing as $\phi' > 0$ or $\phi' < 0$, i.e., as $\phi > .5$ or $\phi < .5$. In particular, if $\phi > .5$, then

$$\phi \otimes min\{\delta_1, \delta_2\} = min\{\phi \otimes \delta_1, \phi \otimes \delta_2\} \text{ and } \phi \otimes max\{\delta_1, \delta_2\} = max\{\phi \otimes \delta_1, \phi \otimes \delta_2\},$$

but if $\phi < .5$, then

$$\phi \otimes min\{\delta_1, \delta_2\} = max\{\phi \otimes \delta_1, \phi \otimes \delta_2\} \text{ and } \phi \otimes max\{\delta_1, \delta_2\} = min\{\phi \otimes \delta_1, \phi \otimes \delta_2\}.$$

This combined with the associativity of $\otimes$ proves

Lemma 4. Let $\mathcal{F}$ and $\mathcal{G}$ be fuzzy formulas, and let ϕ be a fuzzy operator. If $\phi > .5$, then $\phi(\mathcal{F} \wedge \mathcal{G}) = \phi\mathcal{F} \wedge \phi\mathcal{G}$ and $\phi(\mathcal{F} \vee \mathcal{G}) = \phi\mathcal{F} \vee \phi\mathcal{G}$. If $\phi < .5$, then $\phi(\mathcal{F} \wedge \mathcal{G}) = \phi\mathcal{F} \vee \phi\mathcal{G}$ and $\phi(\mathcal{F} \vee \mathcal{G}) = \phi\mathcal{F} \wedge \phi\mathcal{G}$. In particular, every fuzzy formula is equivalent to one in which 1 is the only fuzzy operator applied to non-atomic arguments. □

In view of the lemma, we say that the fuzzy operator ϕ is *positive* or *negative* as $\phi > .5$ or $\phi < .5$. Similarly, we refer to a sign as *positive* or *negative*, if the sign is a subset of, respectively, (.5, 1] or [0, .5). Note that signs may be neither positive nor negative.

3.2 Fuzzy Resolution

The fuzzy logic formulas considered in [19] may be thought of as being in conjunctive normal form in that they are conjunctions of disjunctions. However, the entire formula, each clause, and each literal has a fuzzy operator. (In classical CNF, all negations are at the atomic level.) Nonetheless we refer to such formulas as being in CNF.

Lemma 4 provides a means of determining the truth value of a fuzzy formula under a given interpretation I. In [19], this is accomplished with the *confidence* $\mathcal{C}_I(\mathcal{F})$ of a formula $\mathcal{F}$, which is defined recursively as follows. Let A be an atom, let $\mathcal{F}$ and $\mathcal{G}$ be fuzzy formulas, and let ϕ be a fuzzy operator. Then

1. $\mathcal{C}_I(A) = I(A)$, i.e., the truth value assigned to A by I[7];
2. $\mathcal{C}_I(\phi\mathcal{F}) = \phi \otimes \mathcal{C}_I(\mathcal{F})$;
3. $\mathcal{C}_I(\mathcal{F} \wedge \mathcal{G}) = min\{\mathcal{C}_I(\mathcal{F}), \mathcal{C}_I(\mathcal{G})\}$;
4. $\mathcal{C}_I(\mathcal{F} \vee \mathcal{G}) = max\{\mathcal{C}_I(\mathcal{F}), \mathcal{C}_I(\mathcal{G})\}$.

The fuzzy resolution rule is pretty much the standard resolution rule once we have a notion of satisfiability and know what a complementary pair of literals is. In essence, for an interpretation to satisfy a formula, we want the confidence of the formula under the interpretation to be close to 1. More precisely, we select a *threshold of acceptability* $\Gamma \in [0, 1]$; we assume that $\Gamma > .5$. Then the interpretation I is said to *Γ-satisfy* the formula $\mathcal{F}$ if $\mathcal{C}_I(\mathcal{F}) \geq \Gamma$.

The significance of the threshold can be made clear by looking at some simple examples. Let $\Gamma = .7$ and consider each of the following three formulas: $.8A$,

[7] Interpretations in [19] are limited to assigning 1 or 0 (*true* or *false*) to atoms; this restriction is unnecessary.

$.2A$, and $.6A$. Suppose A is 1; i.e., $I(A) = 1$ for some interpretation I. Then $\mathcal{C}_I(.8A) = .8 \geq \Gamma$, so the first formula is satisfied. The latter two evaluate to .2 and to .6, and so neither is satisfied. Now suppose A is 0. The first formula evaluates to $.8 \otimes 0 = .2$, and the second evaluates to $.2 \otimes 0 = .8$, so the second formula is Γ-satisfied. In effect, since the fuzzy operator .2 is less than $1 - \Gamma$, $.2A$ is a negative literal and is Γ-satisfied by assigning *false* to the atom A. The value of the third formula is now $.6 \otimes 0 = .4$. Thus, in either case, the third formula is Γ-unsatisfiable. The authors of [19] in fact define a clause to be Γ-*empty* if every fuzzy operator of every literal in the clause lies between $1 - \Gamma$ and Γ; it is straightforward to prove that every Γ-empty clause is Γ-unsatisfiable.

The fuzzy resolution rule relies upon complementary pairs of literals. In view of the above comments, we define two literals $\phi_1 A$ and $\phi_2 A$ to be Γ-*complementary* if $\phi_1 \leq 1 - \Gamma$ and $\phi_2 \geq \Gamma$. Fuzzy resolution can now be defined in the obvious way: Let C_1 and C_2 be two clauses, and let L_1 and L_2 be Γ-complementary literals in C_1 and C_2, respectively. Let B_1 and B_2 be the two clauses such that $C_i = B_i \vee L_i, i = 1, 2$. Then the Γ-*resolvent* of C_1 and C_2 on the literals L_1 and L_2 is the clause $B_1 \vee B_2$. In [19] the authors prove the appropriate results, including completeness, for fuzzy resolution.

3.3 Signed Inference with Fuzzy Operator Logic

Let Λ be a fuzzy operator MVL, and let $\mathcal{F}$ be a formula in Λ. By Lemma 4, we may assume that all fuzzy operators in $\mathcal{F}$ are at the atomic level. (More precisely, we are assuming that 1 is the only non-atomic fuzzy operator.) We wish to analyze signed formulas $S\!:\!\mathcal{F}$, where S is any sign, i.e., where $S \subset [0, 1]$. Consider first $\{.5\}\!:\!\mathcal{F}$. For any Λ-consistent interpretation I_{Λ_S} (with corresponding interpretation I over Λ), $I_{\Lambda_S}(\{.5\}\!:\!\mathcal{F}) = \mathit{true}$ iff $I(\mathcal{F}) = .5$. By the observation before Lemma 4, the latter can occur only if some atom in $\mathcal{F}$ is assigned .5 by I or if some fuzzy operator in $\mathcal{F}$ is .5. Both of these conditions are trivial (at least uninteresting), so we shall assume that $.5 \notin S$. The signs in which we are interested in this paper are the *regular signs*; i.e., signs that are intervals containing 0 or 1. (A similar notion was introduced in [5].) Since we are only considering signs that do not contain .5, the regular signs are positive (i.e., subsets of (.5, 1]) or negative (i.e., subsets of (.5, 1]).

To apply any of the inference techniques that have been developed for signed formulas requires driving all signs inward to the atomic level; that is, the inference rules for Λ_S operate on Λ-atomic formulas. Thus, we must determine how to drive positive and negative regular signs to the atomic level. There are three cases to consider: $\mathcal{F} = \mathcal{G} \vee \mathcal{H}$, $\mathcal{F} = \mathcal{G} \wedge \mathcal{H}$, and $\mathcal{F} = \phi A$, where A is an atom. (We need not consider the case $\mathcal{F} = \phi\mathcal{G}$ for an arbitrary $\mathcal{G}$ since we are assuming that the fuzzy operators are all at the atomic level.) The first two cases are handled by the next lemma.

Lemma 5. If S is positive and regular then $S : (\mathcal{G} \vee \mathcal{H}) \equiv_\Lambda S : \mathcal{G} \vee S : \mathcal{H}$ and $S\!:\!(\mathcal{G} \wedge \mathcal{H}) \equiv_\Lambda S\!:\!\mathcal{G} \wedge S\!:\!\mathcal{H}$; if S is negative and regular, then $S\!:\!(\mathcal{G} \vee \mathcal{H}) \equiv_\Lambda S\!:\! \mathcal{G} \wedge S\!:\!\mathcal{H}$, and $S\!:\!(\mathcal{G} \wedge \mathcal{H}) \equiv_\Lambda S\!:\!\mathcal{G} \vee S\!:\!\mathcal{H}$. □

To drive the sign S past the fuzzy operator ϕ in $S\!:\!(\phi A)$, again let $\phi = .5+\phi'$; as before, we have $\Phi_\phi(\delta) = \phi \otimes \delta = .5 - \phi' + 2\phi'\delta$. If $\phi > .5$ (i.e., $\phi' > 0$), then we know that Φ_ϕ is monotonically increasing in δ. Let S be the regular positive sign $[\gamma, 1]$, and let $\Gamma = \Phi_\phi(\gamma)$. Since $\Phi_\phi(.5) = .5$, $\Gamma > .5$, and so $\Phi_\phi(S) = [\Gamma, \phi]$ is a positive sign. In general, unless $\phi = 1$, $\Phi_\phi(S)$ will not be regular. However, $\Phi_\phi^{-1}([\Gamma, 1]) = \Phi_\phi^{-1}([\Gamma, \phi]) = [\gamma, 1]$, which of course is regular and positive, so we have

Lemma 6. Let S be a regular sign and let ϕ be a positive fuzzy operator. Then $\Phi_\phi^{-1}(S)$ is regular and positive or negative as S is positive or negative. If ϕ is negative, then $\Phi_\phi^{-1}(S)$ is regular and positive or negative as S is negative or positive. □

Lemma 7. Let A be an atom in Λ, let S be a (positive or negative) regular sign. Then $S\!:\!\phi A \equiv_\Lambda \Phi_\phi^{-1}(S)\!:\!A$. □

Observe that $\Phi_\phi^{-1}(S)$ will be empty unless there is at least one γ with $\gamma < \phi$ or $\phi < 1-\gamma$ (in the case that S is positive; the reverse must hold if S is negative). In the case when it is empty, then of course $\Phi_\phi^{-1}(S)\!:\!A \equiv_\Lambda \mathit{false}$.

The next theorem is now straightforward.

Theorem 8. Let $\mathcal{F}$ be a fuzzy formula, let S be a positive or negative regular sign, and let $\mathcal{F}'$ be the signed formula that is produced by repeated applications of Lemmas 4 and 5 to $S\!:\!\mathcal{F}$ until all fuzzy operators other than 1 and all signs are at the literal level. Then $S\!:\!\mathcal{F} \equiv_\Lambda \mathcal{G}$ where $\mathcal{G}$ is the formula obtained from $\mathcal{F}'$ by replacing each literal $S\!:\!\phi A$ in $\mathcal{F}'$ with $\Phi_\phi^{-1}(S)\!:\!A$. □

For example, let $\Gamma = .7$, so that $S = [.7, 1]$, and consider the following signed atoms: $S\!:\!.8A$, $S\!:\!.25A$, and $S\!:\!.6A$. The corresponding Λ-atomic atoms are $[.33, 1]\!:\!A$, $[0, .1]\!:\!A$, and $\emptyset\!:\!A$. Notice that $.8A$ and $.25A$ are Γ-complementary, and that the signs of the corresponding signed atoms have an empty intersection. Moreover, the fuzzy literal $.6A$ is Γ-unsatisfiable, and the sign of the corresponding signed atom is empty. It is trivial to verify that these observations generalize to the next lemma.

Lemma 9. Given a threshold of acceptability Γ, let $S = [\Gamma, 1]$, let ϕA, $\phi_1 A$, and $\phi_2 A$ be fuzzy literals with $\phi_1 < \phi_2$, and let $S'\!:\!A$, $S'_1\!:\!A$, and $S'_2\!:\!A$ be the Λ-atomic atoms that correspond, respectively, to $S : \phi A$, $S : \phi_1 A$, and $S : \phi_2 A$. Then $1 - \Gamma < \phi < \Gamma$ iff $S' = \emptyset$. Moreover, $\phi_1 A$, and $\phi_2 A$ are Γ-complementary (i.e., $\phi_1 < \Gamma$ and $\phi_2 > \Gamma$) iff S'_1 and S'_2 are non-empty and $S'_1 \cap S'_2 = \emptyset$. □

The next theorem follows immediately from the last lemma.

Theorem 10. A deduction by fuzzy resolution is a special case of a deduction by signed resolution. □

For the first order case, due to space limitations, we simply state

Theorem 11. Let $(\forall x)\mathcal{F}$ be a fuzzy formula, let I be any interpretation, and let I_{Λ_S} be the corresponding Λ-consistent interpretation. Then I_{Λ_S} maps $(\forall x)\,[\Gamma, 1] : \mathcal{F}$ to *true* iff I Γ-satisfies $(\forall x)\mathcal{F}$, and I_{Λ_S} maps $(\exists x)\,[\Gamma, 1] : \mathcal{F}$ to *true* iff I Γ-satisfies $(\exists x)\mathcal{F}$. □

Examples.

1. Let $C_1 = .2A \vee .9B, C_2 = .98A \vee .87A$, and let $\Gamma = .75$. Then

$$[\Gamma, 1]{:}C_1 \equiv_\Lambda ([0, .083]{:}A) \vee ([.8125, 1]{:}B)$$

and

$$([\Gamma, 1]{:}C_2) \equiv_\Lambda ([.76, 1]{:}A) \vee ([.838, 1]{:}A).$$

Merging the two literals in the second clause and resolving yields $[.8125, 1]{:}B$.

2. Let $C_1 = .11A \vee .43A \vee .19A$, $C_2 = .63B \vee .66A$, and let $\Gamma = .65$. Then

$$[\Gamma, 1]{:}C_1 \equiv_\Lambda ([0, .31]{:}A) \vee (\emptyset{:}A) \vee ([0, .258]{:}A)$$

and

$$[\Gamma, 1]{:}C_2 \equiv_\Lambda (\emptyset{:}B) \vee ([.969, 1]{:}A).$$

Note that the second literal of the first clause and the first literal of the second clause are signed with the empty set. Thus they may be removed without changing the models of the clauses, and we may resolve the other literals to yield the empty clause.

3. Let $C_1 = .1A \vee .2A$, $C_2 = .8A$, and $\Gamma = .7$. Then

$$[\Gamma, 1]{:}C_1 \equiv_\Lambda ([0, .34]{:}A) \vee ([0, .167]{:}A) \text{ and } [\Gamma, 1]{:}\mathrm{C}_2 \equiv_\Lambda [.62, 1]{:}\mathrm{A}.$$

Applying signed resolution produces the empty clause.

4 Summary

The value of our observations on fuzzy operator logic is not so much that the fuzzy resolution rule can be captured by signed resolution. What is potentially most interesting is that, since signed formulas are classical, most classical inference techniques can easily be adapted to signed formulas and hence to fuzzy operator logic. The tableau method and path dissolution, for example, have the property that they produce models of satisfiable formulas. This is a topic of ongoing research that may prove quite useful for some applications. More generally, if one is interested in adapting a particular classical inference technique to fuzzy logic, signed formulas provide an easy method for doing so.

Acknowledgement.

We have benefited from comments offered by Reiner Hähnle on an earlier version of this work.

References

1. Bezdek, J. C., Fuzzy models – what are they, and why?, *IEEE Transactions on Fuzzy Systems*, 1, 1 (1993), 1-6.
2. Blair, H.A. and Subrahmanian, V.S., Paraconsistent logic programming, *Theoretical Computer Science*, 68 (1989), 135-154.
3. Doherty, P., Preliminary report: NM3 — A three-valued non-monotonic formalism. *Proc. of the 5th International Symposium on Methodologies for Intelligent Systems*, (1990), 498-505.
4. Gaines, B. R., Foundations of fuzzy reasoning, *Fuzzy Automata and Decision Processes* (Gupta, M.M., Saridis, G.N., Gaines, B.R eds.), North-Holland (1977), 19-75.
5. Hähnle, R., Uniform notation tableau rules for multiple-valued Logics, *Proc. of the International Symposium on Multiple-Valued Logic*, (1991), 26-29.
6. Hähnle, R., *Automated Proof Search in Multiple-Valued Logics*, Oxford University Press (1993).
7. Kifer, M. and Lozinskii, E., RI: A logic for reasoning with inconsistency, *IEEE Symposium on Logic in Computer Science* (1989), 253-262.
8. Kifer, M. and Lozinskii, E., A logic for reasoning with inconsistency, *Journal of Automated Reasoning* 9 (1992), 179-215.
9. Kifer, M. and Subrahmanian, V.S., Theory of generalized annotated logic programming and its applications, the *Journal of Logic Programming* 12 (1992), 335-367.
10. Lee, R., Fuzzy logic and the resolution principle. *J. ACM* 19 (1972), 109-119.
11. Liu, X.H., Tsai, J.P., Weigert, T., *A*-resolution and the interpretation of *A*-implication in fuzzy operator logic, *Information Science* 56 (1991), 259-278.
12. Lu, J.J., Murray, N.V., Rosenthal, E., Signed formulas and annotated logics, *Proc. of the 23rd IEEE International Symposium on Multiple-Valued Logic*, (1993), 48-53.
13. Mukaidono, M., Fuzzy inference of resolution style, in *Fuzzy Set and Possibility Theory*, R. Yager (Ed.), Pergamon, NY (1982), 224-231.
14. Murray, N.V. and Rosenthal, E., Improving tableaux proofs in multiple-valued logic, *Proc. of the 11th International Symposium on Multiple-Valued Logic* (1991), 230-237.
15. Murray, N. V. and Rosenthal, E., Resolution and path dissolution in multiple-valued logics, *Proc. of the 6thInternational Symposium on Methodologies for Intelligent Systems*, Springer v.542 (1991), 570-579.
16. Murray, N. V. and Rosenthal, E., Signed formulas: a classical approach to multiple-valued logics, TR91-12 (1991), SUNY at Albany. Presented at the 1992 summer meeting of the ASL.
17. Suchón, W., La méthode de Smullyan de construire le calcul n-valent de Lukasiewicz avec implication et négation. Reports on Mathematical Logic, Universities of Cracow and Katowice (1974) 2, 37-42.
18. Surma, S. J., An algorithm for axiomatizing every finite logic. In *Computer Science and Multiple-Valued Logics*, David C. Rine, Ed., North-Holland, Amsterdam, (1984), 143-149.
19. Weigert, T. J., Tsai, J.P., Liu, X.H., Fuzzy operator logic and fuzzy resolution, *J. Automated Reasoning* 10 (1993), 59-78.

Rough Mereology

Lech Polkowski[1] and Andrzej Skowron[2]

[1] Institute of Mathematics, Warsaw University of Technology
Pl. Politechniki 1, 00-650 Warsaw, Poland
e-mail:mltpolk@plwatu21.bitnet
[2] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
e-mail: skowron@mimuw.edu.pl

Abstract. We show how to organize a collection of intelligent cooperating agents into a rough mereological controller for the purpose of assembling a complex object with desired quality from its elementary parts. This scheme for synthesis of a complex object from its elementary parts provides a framework for e.g. quality control in concurrent engineering as well as cooperative problem-solving.

1 Introduction

There are various approaches to the problem of measuring the degree to which elements belong to a given set. Among them are fuzzy set theory [10] and rough set theory [3].

We propose to take as a primitive notion a function (called *a rough inclusion*) returning the degree of inclusion of one set into another. We assume [5] some natural conditions motivated by properties of rough membership functions [4] which characterize rough inclusions. Any rough inclusion can be regarded as the family of partial inclusion relations (indexed by degree of inclusion being the value of the rough inclusion); the relation of partial inclusion in degree α can also be interpreted as the relation of being a part in degree α.

It is natural to require that the relation of being a part in degree 1 satisfies the axioms of mereology, an alternative set theory proposed by Stanislaw Lesniewski [1] originally as a scheme to avoid the Russellian antinomies in naive set theory of Cantor. Our conditions on a rough inclusion guarantee that this demand is fulfilled. We therefore call the theory of rough inclusions rough mereology [5]. There are some motivations for investigating rough mereology, which seem to be important for applications, among them the following.

Rough merology offers a semantics for a higher level language built for the purpose of reasoning about complex structures when it is sufficient to deal only with objects and relations of being a part to a degree; in this case we are free from the implementation details of this language in a lower level language which is more specific and can be based e.g. on standard set theory. This higher level language should be more convenient for applications concerning e.g. assembling of a complex object from elementary parts (concurrent engineering as well as cooperative problem-solving [9]), in particular for quality control [7],[8], or for specifying and analysing of a complex structure where the inference engine must take into account the uncertain character of knowledge about objects and complex structures. Rough mereology permits to consider objects which are not constructible from full parts (parts in degree 1) nevertheless built from objects which are their parts to a certain degree.

The primitive notion of *mereology* [1] is the *relation of being a part* subject to the conditions

1. $part(X,Y) \Rightarrow non[part(Y,X)]$ and
2. $part(X,Y) \wedge part(Y,Z) \Rightarrow part(X,Z)$.

   The secondary relation of *being an ingredient* is defined by

3. $ingredient(X,Y) \Leftrightarrow (part(X,Y) \vee X = Y)$.

Consider a function μ satisfying conditions A-D below on a collection of objects.

A) $\mu(X,Y) \in [0,1]$ for any pair X, Y of objects
B) $\mu(X,X) = 1$ for any object X
C) $[\mu(X,Y) = 1] \Rightarrow [\mu(Z,Y) \geq \mu(Z,X)]$ for any triple X, Y, Z of objects
D) $\exists N \forall X\, \mu(N,X) = 1$.

The symbol $X =_\mu Y$ will stand for $\mu(X,Y) = 1 \wedge \mu(Y,X) = 1$. An object N satisfying D will be called a *null object*. The condition C expresses monotonicity of μ.
We will further require that μ satisfies the conditions

E) if $\forall Z \neq_\mu N[(\mu(Z,Y) = 1) \Rightarrow \exists T \neq_\mu N(\mu(T,Z) = 1 = \mu(T,X)]$ then $\mu(Y,X) = 1$ for any pair X, Y of objects
F) for any collection $\mathcal{U}$ of objects there exists an object X with the following properties
  (i) $\forall Z \neq_\mu N[\mu(Z,X) = 1 \Rightarrow \exists W \neq_\mu N\ \exists T \in \mathcal{U}(\mu(W,Z) = \mu(W,T) = \mu(T,X) = 1]$
  (ii) $\forall Z \in \mathcal{U}[\mu(Z,X) = 1]$
  (iii) $\forall Y[\mu(X,Y) < 1 \Rightarrow$ (either (i) or (ii) does not hold when X is replaced by Y)]

E expresses the inference rule about being a part from being a subpart and F expresses the existence and uniqueness of the object being the class of objects from a given collection.

Functions μ satisfying A-F are called *rough inclusions* [5]. Rough inclusions can be interpreted as hierarchies of relations of being a part in a degree. Rough mereology, the theory of rough inclusion contains mereology of Leśniewski [5,6], (cf. [5, Thms. 3.3.1 - 3.10.2]).

Example 1. An *information system* [3] is a triple A $= (U, A, V)$ where (i) U is a (finite) set called the *universe* (ii) A is a set of *conditional attributes*, any $a \in A$ is a mapping $a : U \longrightarrow V$ with $V_a = a(U)$ - the *value set of a* (iii) $V = \cup\{V_a : a \in A\}$. Any object $x \in U$ is described by its information vector $\mathit{Inf}(x) = \{(a, a(x) : a \in A\}$; objects x and y are indiscernible whenever

$Inf(x) = Inf(y)$. We denote by $INF(x)$ the indiscernibility class containing x. The degree in which objects in U are elements of a given subset $X \subseteq U$ is given by the rough membership function [4] μ_X defined on U by

$$\mu_X(x) = \frac{|\, INF(x) \cap X\,|}{|\, INF(x)\,|}$$

We may observe that the above function μ_X is a particular case of a more general function μ defined on subsets $X, Y \subseteq U$ by $\mu(X,Y) = \frac{|X \cap Y|}{|X|}$ in case $X \neq \emptyset$ and $\mu(\emptyset, Y) = 1$. One may check that μ is a rough inclusion on the powerset of U.

Given μ we define the relation *part* by

1. $part(X,Y)$ iff $\mu(X,Y) = 1 \wedge \mu(Y,X) < 1$ for any pair X, Y of objects and then the relation of *being an ingredient* is defined by
2. $ingr(X,Y)$ iff $\mu(X,Y) = 1$ for any pair X, Y of objects.

We will call a *rough mereological n-connective* ($rmn - c$, in short) any mapping $\mathcal{F} : [0,1]^n \longrightarrow [0,1]$ such that $\mathcal{F}(x_1, x_2, \ldots, x_n) = \mathcal{F}(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$ for any permutation π of $\{1, 2, \ldots, n\}$ (symmetry) and $\mathcal{F}(x_1, x_2, \ldots, x_n)$ is non-decreasing with respect to each coordinate (monotonicity).
The following binary functors of many-valued logic are examples of $rm2 - c's$ (of respectively, Menger, Zadeh, Ruspini): $\mathcal{F}_1(x,y) = x \cdot y$, $\mathcal{F}_2(x,y) = \min\{x,y\}$, $\mathcal{F}_3 = \max\{x + y - 1, 0\}$. Let us note that selecting a binary rmc $\mathcal{F}$ defines a family $\{\tau_\alpha : \alpha \in [0,1]\}$ of tolerance relations where $\tau_\alpha(X,Y)$ iff $\mathcal{F}(\mu(X,Y),$ $\mu(Y,X)) \geq \alpha$. The function $E(X,Y) = \mathcal{F}(\mu(X,Y),\, \mu(Y,X))$ will be called *the mereological closeness function.*

2 Synthesis of complex objects

Synthesis (assembling) of complex objects from parts falls into province of *cooperative product development (concurrent engineering, cooperative problem solving)* [9]. In this many - faceted task one can distinguish few principal aspects. The *spatial aspect* (a *complete process plan*) is represented usually by a hierarchical tree structure *(P-graph)*; the nodes of the tree represent process steps along with information about components, connections, tools and operations. The *communication aspect* involves notification of operators at various nodes of changes in the process (e.g. *constraints adjusting*) and *conflict resolution (negotation)* among operators. The *quality control aspect* [7], [8] deals with routines assuring that the quality of a final assembly assigned by the assembler agrees with the quality decided by a posteriori testing. The *representation aspect* deals with problems of modelling assemblies in terms of objects representing components and connections (the *connection graph*), representing classes of parts by common information *(group technology)*, and representing the *inventory* by means of specifications defining classes of parts as well as by descriptions of physical instances of parts.

Setting an *assembling scheme* as above can be divided into two stages: the *learning stage* and the *production stage*. We describe now a formal rough mereological model of the production stage. The learning stage will be discussed elsewhere.

3 Rough mereological controllers

We describe a system of cooperating intelligent agents acting as a rough mereological controller i.e. it transforms an input (a semantic constraint (specification) which may be expressed in natural language) into an output being the final assembly and its quality value. This system acts on the following principles.

1. The agents are attached to nodes of a tree being a scheme of the assembly process organization. Any agent is provided with an information system. Indiscernibility classes of the system are represented by pattern assemblies. The root agent is provided with a decision system whose decision attribute Q represents the quality value of the final assembly. The leaf agents have an access to the inventory subsets and have a given strategy for computing values of the so called initial rough inclusion μ^0 from their information systems. Any non-leaf agent a has given a set $\mathcal{S}_a$ of assembling procedures, a set $\mathcal{C}_a$ of rough mereological connectives and a mapping $\Phi_a : \mathcal{S}_a \longrightarrow \mathcal{C}_a$.
2. The root agent receives an input being a semantic constraint (specification) on the final assembly expressed as a formula in variables representing conditional attributes of the root agent (or in natural language descriptors of these attributes). The root agent decomposes this input into a constraint sent to its children.
3. The top-down communication consists in decomposing the input constraint at each agent into constraints for its children nodes.
4. Any leaf agent a satisfies its constraint by applying a best-choice strategy leading to selecting a part X from the inventory. The strategy of a is based on μ^0 and consists in covering the space of information vectors $INF(X)$ of inputs by tolerance sets $\tau_\alpha(Y_i)$ (with respect to appropriately chosen tolerance relation τ_α) of information vectors $INF(Y_i)$ of selected pattern parts $Y_1, \ldots, Y_k$. Choosing a part X may consist e.g. in creating the set V of virtual parts (possibly not accessible in the subinventory of a) satisfying the constraint and selecting an object X from the tolerance set $\tau_\alpha(Y_i)$ of a pattern Y_i (possibly chosen at random) closest to V with respect to $E(X,Y)$. The agent a calculates from the initial rough inclusion μ^0 the vector $[E(X,Y_1), \ldots, E(X,Y_k)]$ of rough mereological distances from X to the pattern parts $Y_1, \ldots, Y_k$.
5. The horizontal communication consists in children nodes of a given parent node negotiating their constraints by means of some strategy and finding a conflict resolution i.e. an assignment of a constraint to any of them.
6. The bottom-up communication consists in any child a sending to the parental node the object X assembled at a along with the vector $[E(X,Y_1), \ldots, E(X, Y_k)]$ of rough mereological distances from the assembly X to the pattern assemblies at a.

7. Any parent agent a selects a procedure o from $\mathcal{S}_a$, applies o to objects $X_1, X_2, \ldots, X_n$ sent by its children to assembly $X = o(X_1, X_2, \ldots, X_n)$ and applies $\Phi_a(o)$ to vectors
$$\left[E\left(X_i, Y_1^i\right), \ldots, E\left(X_i, Y_{k_i}^i\right)\right]$$
sent by children to calculate the vector $[E(X, Y_1), \ldots, E(X, Y_k)]$ where Y_1, $Y_2, \ldots, Y_k$ are pattern assemblies at a.
8. The root agent performs Step 7 assembling a final product X and calculating the vector $[E(X, Y_1), \ldots, E(X, Y_k)]$ and applies to this vector some best-fit strategy to find a pattern Y_i^* best-fitting X. Then it outputs the pair $(X, Q(Y_i^*))$.

Let us indicate the main points of this approach.

a) The rule structure of a rough controller is twofold:

(i) the *deep* structure of the controller is expressed in terms of semantic relations among information systems of agents and their children: it may be expressed either as natural language expressions relating conditional attributes of the agent to conditional attributes of its children or as true formulae in variables of conditional attributes of the agent and its children;

(ii) the *surface structure* of the controller is expressed by best-choice strategies of leaf agents, rules of selecting $\Phi_a(o)$ and by algoritms of propagating rough mereological closeness E from the children of an agent to the agent.

b) *Quantization* of fuzzy variables in a fuzzy controller [2] into a number of points corresponding to elements of universe of discourse has its counterpart in the rough mereological controller in covering the space of information vectors of inputs at the root agent by tolerance sets of information vectors of some patterns $Y_1, \ldots, Y_k$.

c) The *fuzzification* process known for fuzzy controllers consists in the rough mereological controller in representing the input at the root agent by the vector $[E(X, Y_1), \ldots, E(X, Y_k)]$ of the values of rough mereological closeness of the final assembly X to the patterns $Y_1, \ldots, Y_k$. This process is realized in a sequence of steps providing the decomposition of the root agent input into inputs for leaf agents, assignment of values of the initial rough inclusion μ^0 and propagation of rough mereological closeness by algorithms based on functions $\mathcal{F}_a$.

d) Control rules of the rough mereological controller are of the form:

(γ) if $\{Y_b : b \text{ a leaf agent}\}$ is a set of patterns selected by leaf agents, then the quality value $Q(Y)$ of the pattern final assembly $Y = class(Y_b : b \text{ a leaf agent})$ is $q \in V_Q$.

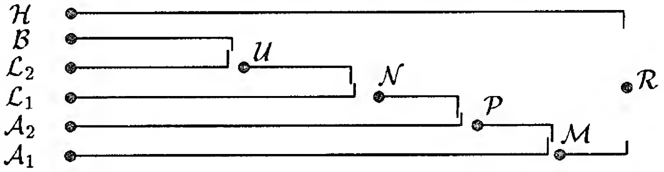
Let us observe that the control rules have a counterpart at each non-leaf agent a in the form of rules like: if $\{Y_b : b \text{ a child of } a\}$ is a set of patterns selected by children of a, then $Y = class(Y_b : b \text{ a child of } a)$ is a pattern at a. These local control rules are encoded in information systems of a and all b relative to inner constraints (see below). Any non-leaf agent may be therefore regarded as a controller on its own.

e) The *defuzzification* process known for fuzzy controllers is realized in the rough mereological controller by assigning to the vector $[E(X, Y_1), \ldots, E(X, Y_k)]$ of **c)** the quality value $Q(X)$.

4 Example

Our example shows the assembling of a complex object (a *gentleman of fortune*) from elementary parts (arms, legs, bodies, heads) and quality control by a rough merological controller.

The system of cooperating intelligent agents



The agent $\mathcal{R}$ (the root agent) produces the final assembly, $\mathcal{M}, \mathcal{P}, \mathcal{N}, \mathcal{U}$ are non-leaf agents submitting auxiliary assemblies and $\mathcal{A}_1, \mathcal{A}_2, \mathcal{L}_1, \mathcal{L}_2, \mathcal{B}, \mathcal{H}$ are leaf agents responsible for submitting elementary parts from the inventory.

We only list leaf agents as other information systems follow by combining pattern objects of leaf agents according to the above scheme relative to deep structure formulae and inner contraints (see below). We also list attribute variables.

$\mathcal{A}_1, \mathcal{A}_2$

Y_{A1}= left normal arm, Y_{A2} = right normal arm, Y_{A3} = hook-ended arm.

	cbuc	wcut	whan	hpis	hrop	mesa	poar	pobj	hcap	hsw	frsa	flsa
Y_{A1}	1	1	1	1	1	1	1	1	1	1	0	1
Y_{A2}	1	1	1	1	1	1	1	1	1	1	1	0
Y_{A3}	1	0	0	0	0	1	0	1	0	0	1	1

cbuc: $x_{A,1}$, *wcut*: $x_{A,2}$, *whan*: $x_{A,3}$, *hpis*: $x_{A,4}$, *hrop*: $x_{A,5}$, *mesa*: $x_{A,6}$, *poar*: $x_{A,7}$, *pobj*: $x_{A,8}$, *hcap*: $x_{A,9}$, *hsw*: $x_{A,10}$, *frsa*: $x_{A,11}$, *flsa*: $x_{A,12}$

$\mathcal{L}_1, \mathcal{L}_2$

Y_{L1} = left normal leg, Y_{L2} = right normal leg, Y_{L3} = strapped wooden leg

	fobj	fpa	fsof	mesl	wou	kck	fcol	fwet	frsl	flsl
Y_{L1}	1	1	1	1	1	1	1	1	0	1
Y_{L2}	1	1	1	1	1	1	1	1	1	0
Y_{L3}	1	0	1	1	0	0	0	0	1	1

fobj: $x_{L,1}$, *fpa*: $x_{L,2}$, *fsof*: $x_{L,3}$, *mesl*: $x_{L,4}$, *wou*: $x_{L,5}$, *kck*: $x_{L,6}$, *fcol*: $x_{L,7}$, *fwet*: $x_{L,8}$, *frsl*: $x_{L,9}$, *flsl*: $x_{L,10}$

$\mathcal{B}$

	pis	cut	bel	kni	par	co	crl	crr
Y_{B1}	1	1	1	1	1	1	1	0
Y_{B2}	1	0	1	0	1	1	0	0
Y_{B3}	0	0	0	1	0	0	0	0
Y_{B4}	1	1	1	0	0	1	0	0

$\mathcal{H}$

	hat	ker	ptc	pig	shd
Y_{H1}	1	0	0	0	0
Y_{H2}	0	0	1	1	0
Y_{H3}	0	1	0	1	0
Y_{H4}	0	1	0	1	0

pis: $x_{B,1}$, *cut:* $x_{B,2}$, *bel:* $x_{B,3}$, *kni:* $x_{B,4}$, *par:* $x_{B,5}$, *co:* $x_{B,6}$, *crl:* $x_{B,7}$, *crr:* $x_{B,8}$, *hat:* $x_{H,1}$, *ker:* $x_{H,2}$, *ptc:* $x_{H,3}$, *pig:* $x_{H,4}$, *shd:* $x_{h,5}$

The meaning of attributes is as follows: *cbuc-carry bucket, wcut-wield cutlass, whan-wield handpike, hpis-hold pistol, hrop-hold rope-end, mesa, mesl-tap message, poar-pull oar, pobj-push objects, hcap-hold capstan bar, hsw-hold steering wheel, frsa, frsl-fit right side, flsa, flsl-fit left side, fobj-feel objects, fpa-feel pain, fsof-feel soft/hard, wou-get wounded, kck-kick, fcol-feel cold/warm, fwet-fel wet/dry, pis-brace of pistols, cut-cutlass, bel-buckled belt, kni-sheathed knife, par-parrot, co-coat, crl-crutch by left, crr-crutch by right, hat-hat, ker-kerchief cover, ptc-eye-patch, pig-tarry pigtail, shd-blind eye shade.* (e.g. Y_{H3} is kerchief-covered head with tarry pigtail and both eyes normal).

The control rules (specifying quality) are:

$Q\ (Y_{B1}Y_{L2}Y_{L3}Y_{A2}Y_{A1}Y_{H1}) = Q(Y_{B1}Y_{L2}Y_{L3}Y_{A2}Y_{A1}Y_{H4}) = LJS$
$Q\ (Y_{B2}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H1}) = CF$
$Q\ (Y_{B2}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H4}) = Q(Y_{B2}Y_{L2}Y_{L1}Y_{A2}Y_{A3}Y_{H1}) =$
$Q\ (Y_{B2}Y_{L2}Y_{L1}Y_{A2}Y_{A3}Y_{H4}) = CE$
$Q\ (Y_{B3}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H2}) = D$
$Q\ (Y_{B3}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H3}) = BP$
$Q\ (Y_{B3}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H4}) = BD$
$Q\ (Y_{B3}Y_{L2}Y_{L1}Y_{A2}Y_{A3}Y_{H4}) = BB$
$Q\ (Y_{B4}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H1}) = Q(Y_{B4}Y_{L2}Y_{L1}Y_{A2}Y_{A1}Y_{H4}) = A$

Q is given by its values on pattern final assemblies.

5 Deep structure

Conditional attributes of any agent a are coded as boolean variables. For a variable $x_{a,i}$, we denote by $\overline{x}_{a,i}$ the expression $\neg x_{a,i}$. An *atomic formula* will be any expression of the form $x_{a,i}$ or $\overline{x}_{a,i}$. A *well - formed formula* χ will be any expression χ belonging to the smallest set of expressions containing all atomic formulae and closed under propositional connectives $\vee$ and $\wedge$. A well - formed formula χ will be called *proper* if there is no variable $x_{a,i}$ such that both $x_{a,i}$ and $\overline{x}_{a,i}$ are among variables of χ. The symbol $[\chi]_v$ denotes the logical value of a well-formed formula χ under a valuation v on its variables where valuations are induced by objects i.e. for an object X, the valuation $v = v_X$ is given by $v_X(x_{a,i}) = x_{a,i}(X)$. A well - formed formula χ is said to be *true* if $[\chi]_v = 1$ for any object - induced valuation v. We say that an object (assembly) X *satisfies a well-formed formula* χ if $[\chi]_{v_X} = 1$ and we write in this case $X \models \chi$.

The following true formulae render the description of the deep structure on the level of consecutive non-leaf agents.

$x_{U,1} \Leftrightarrow x_{B,1} \wedge x_{B,2} \wedge x_{B,4}$ ($x_{U,1}$ means *heavily armed*: pistols, cutlass, knife)

$x_{U,2} \Leftrightarrow (x_{B,1} \wedge x_{B,2} \wedge \overline{x}_{B,4} \vee x_{B,2} \wedge \overline{x}_{B,1} \wedge x_{B,4} \vee x_{B,4} \wedge x_{B,1} \wedge \overline{x}_{B,2})$

$x_{U,3} \Leftrightarrow (x_{B,1} \wedge \overline{x}_{B,2} \wedge \overline{x}_{B,4} \vee x_{B,2} \wedge \overline{x}_{B,1} \wedge \overline{x}_{B,4} \vee x_{B,4} \wedge \overline{x}_{B,1} \wedge \overline{x}_{B,2})$

$x_{U,4} \Leftrightarrow x_{B,7}, x_{U,5} \Leftrightarrow x_{B,8}, x_{U,6} \Leftrightarrow x_{B,5}, x_{U,7} \Leftrightarrow x_{B,6}$

$x_{N,1} \Leftrightarrow x_{U,1}, x_{N,2} \Leftrightarrow x_{U,2}, x_{N,3} \Leftrightarrow x_{U,3}, x_{N,4} \Leftrightarrow x_{U,4} \vee x_{U,5}$

$x_{N,5} \Leftrightarrow x_{U,6}, x_{N,6} \Leftrightarrow x_{U,7}$

$x_{P,1} \Leftrightarrow x_{N,1}, x_{P,2} \Leftrightarrow x_{N,2}, x_{P,3} \Leftrightarrow x_{N,3}, x_{P,4} \Leftrightarrow x_{N,4},$

$x_{P,5} \Leftrightarrow x_{N,5} \vee x_{N,6}, x_{P,6} \Leftrightarrow x_{A,11} \wedge x_{A,2}$

$x_{P,1} \Leftrightarrow x_{M,1}, x_{P,2} \Leftrightarrow x_{M,2}, x_{P,3} \Leftrightarrow x_{M,3}, x_{P,4} \Leftrightarrow x_{M,4}, x_{P,5} \Leftrightarrow x_{M,5}$

$x_{M,6} \Leftrightarrow x_{P,6} \vee (x_{A,12} \wedge \overline{x}_{A,2})$

$x_{R,1} \Leftrightarrow x_{M,1}, x_{R,2} \Leftrightarrow x_{M,2}, x_{R,3} \Leftrightarrow x_{M,3}, x_{R,4} \Leftrightarrow x_{M,4}, x_{R,5} \Leftrightarrow x_{M,5}$

$x_{R,6} \Leftrightarrow x_{M,6}, x_{R,7} \Leftrightarrow x_{H,1} \vee x_{H,2}, x_{R,8} \Leftrightarrow x_{H,4}$

$x_{R,9} \Leftrightarrow \overline{x}_{M,4} \wedge \overline{x}_{M,5} \wedge \overline{x}_{M,6} \wedge \overline{x}_{H,1}$

$x_{R,10} \Leftrightarrow \overline{x}_{M,4} \wedge \overline{x}_{M,5} \wedge \overline{x}_{M,6} \wedge \overline{x}_{H,1} \wedge \overline{x}_{H,3} \wedge \overline{x}_{H,5} \wedge x_{M,3}$

$x_{R,11} \Leftrightarrow \overline{x}_{M,5} \wedge \overline{x}_{M,6} \wedge \overline{x}_{H,1}$

6 Logic of constraint negotiation and conflict resolution

We distinguish between *inner constraints* imposed on agents by the designer independently of a particular assembling task and *outer constraints* issued in the particular assembling process. Inner constraints can be interpreted as *meaning postulates in the sense of Carnap* and outer constraints express the particular demand on the final assembly. The constraints may be expressed as natural language phrases and then rendered into formulae in the corresponding variables. For constraints $\chi_1, \chi_2, \ldots, \chi_k$ at a, the *negotiation function* is

$$f^a_{\chi_1,\chi_2,\ldots,\chi_k} = [\chi_1] \wedge [\chi_2] \wedge \ldots \wedge [\chi_k]$$

where $[\chi_i]$ is the formula χ_i in which each variable $x_{a,j}$ has been replaced by its deep structure rendering in terms of variables of the children of a.

Let $\bigvee \bigwedge (x'_{a,i})$ be the normal disjunctive form of $f^a_{\chi_1,\chi_2,\ldots,\chi_k}$. Then any object X such that $X \models f^a_{\chi_1,\chi_2,\ldots,\chi_k}$ satisfies $X \models \bigwedge(x'_{a,i})$ for some proper prime implicant $\bigwedge(x'_{a,i})$ of $f^a_{\chi_1,\chi_2,\ldots,\chi_k}$. Our negotiation and conflict resolution strategy will therefore consist of the two following steps at a.

Step 1. A proper prime implicant $\bigwedge(x'_{a,i})$ of the negotiation function $f^a_{c_1,\chi_2,\ldots,\chi_k}$ is selected. This prime implicant is a constraint (conflict) resolution at a.

Step 2. Let $\bigwedge(x'_{a,i}) = \chi_{b_1} \wedge \chi_{b_2} \wedge \ldots \wedge \chi_{b_k}$ where $b_1, b_2, \ldots, b_k$ are the children of a and any χ_{b_i} is a conjunction of variables of b_i. Then the constraint χ_{b_i} is sent to b_i. In case b_i is a leaf agent, the condition χ_{b_i} is resolvable if there exists a part X accessible by the agent b_i i.e. $\tau_\alpha(X, Y^*)$ where Y^* is a pattern part with $E(Y^*, V) = \min\{E(Y, V) :$ a pattern $Y\}$ and $V = \{X' : X'$ is a virtual part such that $X' \models \chi_{b_1}\}$; such X is a constraint resolution at b_i. We first describe an exemplary set of inner constraints:

at $\mathcal{A}_1 : x_{A,12}$, at $\mathcal{A}_2 : x_{A,11}$, at $\mathcal{L}_1 : x_{L,10}$, at $\mathcal{L}_2 : x_{L,9}$, at $\mathcal{B} : \overline{x}_{B,8} \vee \overline{x}_{B,7}$, at $\mathcal{U} : x_{B,8} \vee x_{L,2} \wedge x_{L,9}, \overline{x}_{B,8} \vee \overline{x}_{L,2} \wedge x_{L,9}$, at $\mathcal{N} : \overline{x}_{U,4} \vee \overline{x}_{L,2} \wedge x_{L,10}, x_{U,4} \vee x_{L,2} \wedge x_{L,10}, \overline{x}_{U,4} \vee \overline{x}_{U,5}$, at $\mathcal{P} : \overline{x}_{N,1} \wedge \overline{x}_{N,2} \vee x_{A,2} \wedge x_{A,11}$, at $\mathcal{M} : \overline{x}_{P,1} \wedge \overline{x}_{P,2} \wedge \overline{x}_{P,6} \vee x_{A,2} \wedge x_{A,12}$, at $\mathcal{R} : \overline{x}_{M,4} \wedge \overline{x}_{M,6} \vee \overline{x}_{H,3} \wedge \overline{x}_{H,5}, \overline{x}_{M,1} \wedge \overline{x}_{M,2} \vee \overline{x}_{H,5}, \overline{x}_{M,5} \vee \overline{x}_{H,5} \wedge (x_{H,1} \vee x_{H,2}), x_{M,5} \vee \overline{x}_{H,1}$

Let the outer constraint at $\mathcal{R}$ be

$(1, \mathcal{R})$ $x_{R,1} \wedge x_{R,4} \wedge x_{R,5} \wedge x_{R,7} \wedge \overline{x}_{R,11}$

The proper prime implicants of the negotiation function f^R are

$(1, \mathcal{R}, \mathcal{H})$ $x_{M,1} \wedge x_{M,4} \wedge x_{M,5} \wedge x_{H,2} \wedge \overline{x}_{H,1} \wedge \overline{x}_{H,3} \wedge \overline{x}_{H,5}$
$(2, \mathcal{R}, \mathcal{H})$ $x_{M,1} \wedge x_{M,4} \wedge x_{M,5} \wedge x_{H,2} \wedge x_{H,1} \wedge \overline{x}_{H,3} \wedge \overline{x}_{H,5}$
$(3, \mathcal{R}, \mathcal{H})$ $x_{M,1} \wedge x_{M,4} \wedge x_{M,5} \wedge x_{H,2} \wedge \overline{x}_{H,3} \wedge \overline{x}_{H,5}$

Let the constraints negotiated by $\mathcal{M}$ and $\mathcal{H}$ be

$(1, \mathcal{H})$ $x_{H,2} \wedge \overline{x}_{H,1} \wedge \overline{x}_{H,3} \wedge \overline{x}_{H,5}$
$(1, \mathcal{M})$ $x_{M,1} \wedge x_{M,4} \wedge x_{M,5}$

Continuing, we get constraints for other leaf agents:

$(1, \mathcal{A}_2)$ $x_{A,2} \wedge x_{A,11}$
$(1, \mathcal{L}_1)$ $x_{L,10} \wedge \overline{x}_{L,2}$
$(1, \mathcal{L}_2)$ $x_{L,2} \wedge x_{L,9}$
$(1, \mathcal{B})$ $x_{B,1} \wedge x_{B,2} \wedge x_{B,4} \wedge x_{B,5} \wedge x_{B,7} \wedge \overline{x}_{B,8}$

The constraint resolution process leads in the case outlined above to the final assembly $X = Y_{L2} Y_{B1} Y_{L3} Y_{A2} Y_{A1} Y_{H4}$. One may check that X is the only positive outcome of the input $(1, \mathcal{R})$.

We will present here the bottom - up communication process. In this process any child sends to its parent a two-component messsage, the first component is an assembly meeting the negotiated solution to constraints and the second component is the vector of values of rough merological closeness between the assembly and any of pattern objects.

We will adopt the following strategy for defining μ^0. For any leaf agent, $\mu^0(X, Y)$ is calculated from the information system describing elementary parts to which the agent has an access by the formula

$$\mu^0(X, Y) = \frac{|\mathit{IND}(X, Y)|}{|A|}$$

where A is the attribute set and $\mathit{IND}(X, Y) = \{a \in A : a(X) = a(Y)\}$.

We will adopt the strategy that takes the rough mereological connective $\mathcal{F}(x, y) = \min\{x, y\}$; accordingly, we have $E(X, Y) = \min\{\mu(X, Y), \mu(Y, X)\}$.

Any agent, given a pattern $Y = Y_1 Y_2$ and an assembly $X = X_1 X_2$ along with values $E(X_1, Y_1)$, $E(X_2, Y_2)$ submitted by children calculates the mereological distance $E(X, Y) = \mathcal{F}(E(X_1, Y_1), E(X_2, Y_2))$. The only exception to this rule is

the following:
the agent $\mathcal{M}$ assigns $E(X,Y)=1$ to any pair X,Y of objects if X and Y satisfy one of the following constraints

(i) $x_{M,3} \wedge x_{M,5} \wedge x_{M,6}$
(ii) $x_{M,3} \wedge \overline{x}_{M,5} \wedge x_{M,6}$

(expressing the identity of symmetrical objects).

The root agent calculates values of $E(X,Y)$ and selects at random the pattern Y^* satisfying $1-E(X,Y^*) = \min\{1-E(X,Y):$ any pattern $Y\}$. Then it assigns the quality of Y^* as the quality of X.

Any leaf agent a selects a part X such that $\tau_{0.85}(X,Y)$ where Y is a pattern part satisfying the constraint. For example $[\mu^0(Y_{L2}, Y_{Li})] = [0.8, 1, 0.4]$.

For the constraint $(1,\mathcal{R})$ the root agent applies his best fit strategy to (X, [0.4, 1, 0.33, 0.4, 0.33, 0.33, 0.25, 0.25, 0.25, 0.4, 0.4]); and selects $Y^* = Y_{B1}Y_{L2}Y_{L3}Y_{A2}Y_{A1}Y_{H4}$. Accordingly, $Q(X) = LJS$.

This work was partially supported by the grant 8-S503-019-06 from State Committee for Scientific Research (Komitet Badań Naukowych).

References

1. Lesniewski, S.: Foundations of the general theory of sets (in Polish), Moscow, 1916; engl. transl. in:Surma, Srzednicki, Barnett, Rickey (eds.), Stanislaw Lesniewski, Collected Works, Kluwer, Dordrecht (1992).
2. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies, **7** (1975) 1–13.
3. Pawlak, Z.: Rough Sets:Theoretical Aspects of Reasoning about Data, Kluwer, Dordrecht (1991).
4. Pawlak, Z., Skowron, A.: Rough membership functions in: Fedrizzi, M., Kacprzyk, J., Yager, R.R., eds., Advances in the Dempster-Shafer Theory of Evidence, Wiley, New York, (1994), 251–271.
5. Polkowski, L., Skowron, A.: Rough mereology, ICS Research Report **11**/94, Institute of Computer Sci., Warsaw Univ. Technology, (1994).
6. Polkowski, L., Skowron, A.: Logic of rough inclusion. Rough mereological functions. Rough functions, ICS Research Report **12**/94, Institute of Computer Sci., Warsaw Univ. Technology, (1994).
7. Pyzdek, T., Berger, R.: Quality Engineering Handbook, ASQC Quality Press, Milwaukee, (1991).
8. Rembold, U., Nnaji, B.O.: Storr,A.,Computer Integrated Manufacturing and Engineering, Addison-Wesley, Wokingham, (1993).
9. Sriram, D., Logcher, R., Fukuda, S.: Computer - Aided Cooperative Product Development, LNCS **492**, Springer-Verlag, Berlin Heidelberg, (1991).
10. Zadeh, L.: Fuzzy sets, Information and Control, **8**(1965), 338–353.

A New Rule for Updating Evidence

ShengLi Shi[1] and M. E. C. Hull[1] and D. A. Bell[2]

[1] Department of Computing Science
[2] Department of Information Systems
University of Ulster
Northern Ireland, BT37 0QB, U.K.

Abstract. In probabilistic reasoning when we have to build a probability distribution on some events and no further information is available, we then apply the *insufficient reasoning principle*; i.e. we assume that the events have equal opportunities to occur. In our previous work we have observed that in the Dempster-Shafer theory of evidence the principle is not applied when using Dempster's rule of combination to combine evidence. In this paper we propose a new rule, *proportional sum*, for updating evidence. A comparison of this rule with Dempster's rule of combination shows that the new rule updates evidence, corresponding to the point of view of belief as generalized probability, while Dempster's rule of combination combines evidence, corresponding to the point of view of belief as evidence as indicated by Halpern and Fagin. Then we have two rules within one framework: one is for combining evidence and another is for updating evidence.

1 Introduction

The Dempster-Shafer theory of evidence[13][9] (the D-S theory for short) generalizes the Bayesian-based model and gives an applicable approach to inference with uncertainty in AI[1]. There has been some criticism of the D-S theory and some "counter-examples" have been proposed[18][11]. Dempster's rule of combination[13], an important tool for combining evidence, has been singled out for criticism. One conclusion, given by Halpern and Fagin[11], is that the confusion regarding the interpretation of the theory results from "having" two viewpoints of belief which result in two different methods to deal with evidence. The first viewpoint is of the D-S theory as a generalization of Bayesian statistics, where the belief is given as *generalized objective probability.* The second is as a way of representing evidence, where the belief represents *credibility.*

There is a well-known principle called the *insufficient reasoning principle* in probability theory. When we have to build a probability distribution on some events and no other information is available, we assume that the events have equal opportunities to occur. An application of this principle is to conditional probability. The motivation for our work comes from the fact that, in the D-S theory the principle is not applied by Dempster's rule of combination: To combine two pieces of evidence, if the intersection of two subsets is not empty, then the masses of these two subsets are totally transferred to the intersection. This contrasts with conditional probability. As Fagin and Halpern indicate in [11], Dempster's rule corresponds to combining evidence, while conditional probability corresponds to updating evidence. The two viewpoints of belief comes from that whether updating or combining evidence. Therefore Dempster's rule does not agree with the Bayesian conditional probability for some examples[14].

In [15] we have observed that the insufficient reasoning principle is an essential concept which dominates the rules for updating or combining evidence. Based on the above observation in this paper we propose a new rule, *proportional sum*, for updating evidence. A comparison of this rule with Dempster's rule of combination shows that the new rule updates evidence which corresponds to the point of view of belief as generalized probability, while Dempster's rule of combination combines evidence which corresponds to the point of view of belief as evidence as indicated by Halpern and Fagin in [11]. Then we have two rules within one framework : one is for combining evidence and another is for updating evidence.

The rest of this paper is organized as follows. In next section we give a brief description of the D-S theory. In section 3 we discuss the reason why Dempster's rule does not agree with Bayesian analysis. In section 4 we propose a new rule for combining evidence based on the insufficient reasoning principle and compare it with Dempster's rule.

2 Representing belief

Let Θ be a finite non-empty set called a *frame of discernment.* Informally, it is a set of all possible truth-values that some propositions of interest can take. Each proposition is associated to a subset of the frame of discernment. If a subset of a frame is considered as a set of *truth-points* taken by the associated proposition, then the subset represents a *proof* of the related proposition, where each truth-point is an element of the proof. There is a correspondence between a proof and a proposition. Therefore we may consider *a proof as a proposition*[3][3]. In our interpretation a proof corresponding to a proposition is represented by a subset of a frame of discernment. Because a proof may not prove a proposition completely, uncertainty arises. From the relationship of propositions and proofs we have subjective beliefs for propositions. This gives a natural interpretation for the D-S theory [17][16].

When we use "propositions" (in the meaning above) to represent uncertain knowledge, a proof of a proposition only proves the proposition to a certain degree. To express this a *mass* of a proposition is introduced, which represents the *weightiness* of a proof of the proposition. The masses of all propositions are given by a *mass function.*

Definition 1. Let Θ be a frame of discernment. A *mass function* m Θ is a mapping from 2^{Θ} to the real closed interval [0,1] with the following restriction:

$$m(\emptyset) = 0, \quad \sum_{A \in 2^{\Theta}} m(A) = 1$$

[3] This viewpoint is known as *Hoard-Curry isomorphism* among logicians.

If a proposition whose proof has non-zero mass related to a mass function then the proposition is said to be a *focal proof* (or *focal element*) (of the mass function). The union of all focal proofs is called *core* of the mass function. When we present a mass function we only give values to focal proofs. Because each subset is a proof corresponding to a proposition, all propositions are related together by the restriction in Definition 1.

Suppose Θ is a frame of discernment. Let m be a mass function defined on Θ. We define $\mathcal{P}(m)$ to be the set of all focal elements (or propositions) of m, i.e.

$$\mathcal{P}(m) = \{A \mid A \in 2^{\Theta} \text{ and } m(A) > 0\}$$

Based on the insufficient reasoning principle, we assume that, all truth-points in a proof of a proposition *prove* the proposition to the same degree. That is, suppose A is a proof of a proposition with mass $m(A)$, then for each $B \subseteq A$ it will prove the proposition with mass $m(A) \cdot |B|/|A|$, where $|B|$ is the cardinal of B (In fact B as a subset of a frame discernment also has a mass. It is a *sub-proof* of A and shares some truth-points with A).

Some other functions are used when conducting reasoning with uncertainty. Examples are the belief function and the plausibility function. All these functions are called *evidential functions.*

Let A and B be two focal proofs. If $B \subseteq A$ then all the truth-points that B can take can also be taken by A as well. Then B also *proves* the proposition represented by A but not vice-versa. We call B a *sub-proof* of A. Now suppose we have a mass function m on a frame of discernment. Given a proof A of a proposition how can we express our degree of belief in the proposition? Actually all sub-proofs of A also prove A though to different degree with different masses. So we should *accumulate* all such proofs to express our degree of belief in A. This is the way that *belief function* and *plausible function* are defined.

Definition 2. Let Θ be a frame and m be a mass function on Θ. The belief function *bel* and plausibility function *pls* induced by m are defined as follows. For each proposition A of Θ

$$bel(A) = \sum_{B \subseteq A} m(B) \quad pls(A) = 1 - bel(\overline{A}) = 1 - \sum_{B \subseteq \overline{A}} m(B)$$

Intuitively, a belief function defines a lower bound on the uncertainty of a proposition; while the corresponding plausibility function gives a upper bound.

Given two mass functions on a frame which stem from two pieces of evidence we need a method to combine them, i.e. to combine two mass functions so that we can get new, combined degrees of belief. Dempster's rule gives a way to combine two mass functions. First we define conditions under which two mass functions can be combined.

Let Θ be a frame and m_1 and m_2 be two mass functions on Θ. If $\sum_{B \cap A \neq \emptyset} m_1(A) \cdot m_2(B) \neq 0$ then m_1 and m_2 are said to be *combinable.*

Definition 3. Let m_1 and m_2 be two combinable mass functions on frame of discernment Θ. *Dempster's rule of combination* $\oplus$ is defined as follows. For each subset $A \neq \emptyset$ of Θ,

$$(m_1 \oplus m_2)(A) = \frac{\sum_{A_i,B_j \subseteq \Theta; A_i \cap B_j = A} m_1(A_i) \cdot m_2(B_j)}{\sum_{A_i,B_j \subseteq \Theta; A_i \cap B_j \neq \emptyset} m_1(A_i) \cdot m_2(B_j)}$$

3 Reasoning when lacking information

In probability theory we apply the *insufficient reasoning principle* when we have to build a probability distribution on some events if no further information exists to indicate it. In such cases we shall assume these events have equal opportunities to occur. The basic idea of this principle is as follows.

> *Suppose there are m events and that the probability of these events collectively occurring is p. If there is no other information about these events, we assume that these events occur with equal probabilities, i.e. we assign an equal probability p/m to each event.*

For example, suppose we have m events with probabilities p_1,...,p_m. If we have a new piece of evidence that these events as a collection occur with a probability p then we assign probabilities to each event according to a proportion of its probability with others. If the events are independent of each other, then we assign $p_i \cdot (p/\sum_j p_j)$ to the ith events.

Example 1. Suppose there are only two events, represented as subsets A and B of a finite universe U with probabilities $p(A)$ and $p(B)$ respectively. Now suppose we obtain a new piece of evidence that event C occurs, i.e. $q(C) = 1$, where q is a probability measure function. Suppose $A \subset C$ and $B \not\subset C$ but $B \cap C \neq \emptyset$, then one problem arises: how do we modify the previous probability? For simplicity we assume $A \cap B = \emptyset$.

A Bayesian, i.e. someone who will apply the insufficient reasoning principle, will probably solve this problem as follows. If D is measurable with probability $p(D)$, then the updated probability distribution p' of A and D with q is defined as follows:

$$p'(A) = p(A)/(p(A) + p(D)) \qquad p'(B) = p(D)/(p(A) + p(D))$$

This result is obtained directly by applying the insufficient reasoning principle.

On the other hand, if D is nonmeasurable, i.e. there is no probability assignment to D, then by the insufficient reasoning principle all sample points in B have the same opportunities to occur. We then extend probability p to p'' such that $p''(A) = p(A)$ and $p''(D) = p(B) \times |D|/|B|$. The extension of p to p'' is reasonable: we just assume equal opportunities of the sample points of B. Now replacing p with p'' we obtain the modified probability p'.

$$p'(A) = p''(A)/(p''(A) + p''(D)) \qquad p'(D) = p''(D)/(p''(A) + p''(D))$$

The corresponding solution of the problem in the D-S theory is as follows. A and B are only focal elements of mass function p. The new piece of evidence is represented as q and C is its only focal element. If we apply Dempster's rule to solve it, then we have

$$(p \oplus q)(A) = \frac{\sum_{X \cap Y = A} p(X) \cdot q(Y)}{\sum_{X \cap Y \neq \emptyset} p(X) \cdot q(Y)} = p(A) \cdot q(C) = p(A)$$

$$(p \oplus q)(D) = p(B) \cdot q(C) = p(B)$$

□

This example shows the difference between the two approaches. (1) Using Dempster's rule to combine two beliefs some new focal proofs may be produced. In the above example D is a focal proof of $p \oplus q$ but is not a focal proof of either p or q. In contrast, using conditional probability there is no new focal proof produced. (2) The insufficient reasoning principle is not applied by Dempster's rule: All the mass of B and C have been transferred to $D = C \cap B$, while in Bayesian method only a part of the probability (or mass) corresponding to $D = C \cap B$ is assigned to all of B rather than just D.

More generally, when one agent's belief is *updated* by another agent's belief, this means that all focal proofs related to the first agent remain unchanged, but the degrees of belief of the propositions might be modified and, therefore some focal proofs may not be focussed since their masses may be zero. On the other hand, when *combining* two agents' beliefs, this means a new set of focal proofs is obtained (e.g. in above example D is obtained), which is a compromised result of two agents, and the degrees of belief of focal proofs are also changed.

The difference between Bayesian probability theory and the D-S theory shown in above example is significant: If we admit that conditional probability is corresponding to *updating* evidence and Dempster's rule to *combining* evidence, then the method of how to assign probabilities (or masses) is essential.

We now consider the well-known *three prisoners problem*. The result is different using the D-S theory and Bayesian analysis. This example is considered in [8][4] and Fagin and Halpern discuss it in [6] and solve it using a new approach to updating beliefs based on inner and outer probability.

Example 2. There are three prisoners a, b and c. Two of them are to be executed but a does not know which. He therefore says to the jailer, "Since either b or c is certainly going to be executed, you will give me no information about my own chances if you give me the name of one man, either b or c, who is going to be executed." Accepting this argument, the jailer truthfully replies, "b will be executed." Thereupon a feels happier because before the jailer replied, his own chance of not being executed was 1/3, but afterwards there are only two people, himself and c, who could be the one not executed, and his chance of execution is 1/2. Is his reasoning correct?

A solution to the problem is as follows. We use a pair (x, y), where $x, y \in \{a, b, c\}$, to model a possible situation in which x is pardoned and the jailer says that y is going to be executed in response to a's question. Then we have $x \neq y$ and $y \neq a$. Thus the possible set is $\{(a, b), (a, c), (b, c), (c, b)\}$. The events that a lives, b lives and c lives, denoted $lives - a$, $lives - b$ and $lives - c$ respectively, correspond to the sets $\{(a,b),(a,c)\}$, $\{(b,c)\}$ and $\{(c,b)\}$ respectively, and each of them has probability 1/3. The set for $says-b$ is $\{(a,b),(c,b)\}$.

For computing $p(lives-a|says-b)$ we need to know $p(lives-a\cap says-b)$, i.e. $p(\{(a,b)\})$. We only know $p(\{(a,b),(a,c)\})$. By the insufficient reasoning principle $\{(a,b)\}$ and $\{(a,c)\}$ have the same opportunities to occur. Then we have $p(\{(a,b)\}) = 1/3 \times 1/2 = 1/6$, and $p(lives-a|says-b) = p(lives-a\cap says-b)/p(says-b) = (1/6)/(1/2) = 1/3$. This means that the jailer's answer does not affect a's probability.

Now we consider this problem in the framework of the D-S theory. We have the frame of discernment $\Theta = \{(a,b),(a,c),(b,c),(c,b)\}$ and mass function m_1: $m_1(\{(a,b),(a,c)\}) = m_1(\{(b,c)\}) = m_1(\{(c,b)\}) = 1/3$.

The jailer's answer gives a new piece of evidence represented as mass function m_2: $m_2(\{(a,b),(c,b)\}) = 1$. But the combination of the two pieces of evidence would not give a correct answer (according to Bayesian): $(m_1\oplus m_2)(\{(a,b)\}) = (m_1\oplus m_2)(\{(c,b)\}) = 1/2$, because the mass of $\{(a,b),(a,c)\}$ has been transferred to $\{(a,b)\}$.

□

4 A Rule for updating evidence

As we indicated in the previous section the insufficient reasoning principle is not applied by Dempster's rule. Based on the insufficient reasoning principle we propose a new rule, called *proportional sum*, for updating evidence.

Given a probability function μ of a probability space $< M, S, \mu >$, if every subset of M is measurable, then μ is said to be a *complete measure.* Generally μ is not defined on 2^M everywhere and some subsets of 2^M might be nonmeasurable, i.e. $2^M \neq S$. There are several ways to extend μ to 2^M. Traditionally μ is extended to the *inner measure* μ_* and *outer measure* μ^* as follows [10]: For any subsets $A \in 2^M$,

$$\mu_*(A) = sup\{\mu(X)|X \subseteq A \text{ and } X \in S\}$$

$$\mu^*(A) = inf\{\mu(X)|A \subseteq X \text{ and } X \in S\}$$

Fagin and Halpern show that every inner measure is a belief function and that every belief function can be viewed as an inner measure[7]. For the extension of a probability space, there is a well-known result [6].

Theorem 1. Let $< M, S', \mu' >$ be an extension of probability space $< M, S, \mu >$, i.e. $S \subseteq S'$ and for all $A \in S$, $\mu'(A) = \mu(A)$. Then for any μ'-measurable $A \in S'$

$$\mu_*(A) \leq \mu'(A) \leq \mu^*(A)$$

We now give another extension to the concept of a probability space based on insufficient reasoning principle.

Definition 4. Given a probability space $< M, S, \mu >$, the *proportional extension* μ^+ of measure μ is constructed as follows: If A is measurable and A does not contain any measurable subsets, then for any subset B of A

$$\mu^+(B) = \mu(A) \cdot |B|/|A|$$

Theorem 2. μ^+ is a complete probability function.

Corollary 3. $\mu_*(A) \leq \mu^+(A) \leq \mu^*(A)$.

Now we consider the application of the above extension to the D-S theory. All mass functions are *complete* in the sense that they are defined on the whole evidential space. From the analysis in section 3, when two pieces are combined, we wish to use proportions of masses of the first piece of evidence rather than whole masses. Therefore the first piece of evidence is *updated* by the second one.

Consider again the case of Example 1. Instead of transferring all the mass of B to $D = C \cap B$, we just reallocate a proportional mass of B corresponding to that of D to D, i.e. a fraction $|D|/|B|$ of the mass of B. In the spirit of the proportional extension of a probability measure, we can assign $p(B) \cdot (|D|/|B|) \cdot q(C)$ to D. This suggests a new rule for updating (rather than combining) evidence in the D-S theory. Since it uses a proportion to represent a combined mass, we call the new rule *proportional sum.* First we define the condition under which one mass function can be updated by another.

Definition 5. Let Θ be a frame of discernment. Suppose that m_1 and m_2 are two mass functions on Θ. Then m_1 is *updatable* by m_2 if

$$\sum_{A_i \in \mathcal{P}(m_1), B_j \in \mathcal{P}(m_2)} m_1(A_i) \cdot \frac{|A_i \cap B_j|}{|A_i|} \cdot m_2(B_j) > 0$$

Definition 6. Let Θ be a frame of discernment. Suppose that m_1 and m_2 are two mass functions on Θ and m_1 is updatable by m_2. Then the *proportional rule of updating* m_1 *with* m_2 $m_1 \oslash m_2$ is defined as follows: For all $A \in \mathcal{P}(m_1)$,

$$(m_1 \oslash m_2)(A) = \frac{m_1(A)/|A| \cdot \sum_{B_j \in \mathcal{P}(m_2)} |A \cap B_j| \cdot m_2(B_j)}{\sum_{A_i \in \mathcal{P}(m_1), B_j \in \mathcal{P}(m_2)} m_1(A_i) \cdot \frac{|A_i \cap B_j|}{|A_i|} \cdot m_2(B_j)}$$

The intuitive idea of the proportional sum is that, when updating a mass function, we only consider the masses of subsets remaining and reallocate masses according to the insufficient reasoning principle. We consider Example 2 again using the new rule of updating evidence.

Example 3. In this example we have one agent (prisoner) a. His initial belief about his chance of living is given by his assumption that the three prisoners have equal opportunities to be pardoned or to be executed. Now the new piece of evidence becomes available, i.e. prisoner b will be executed. Here a ought to update his belief with the new piece of evidence rather than use combination[4]. Now we use the proportional sum to combine mass functions m_1 and m_2.

$$(m_1 \oslash m_2)(\{(a, b)\}) = (1 \times \frac{1}{3} \times \frac{1}{2})/(\frac{1}{2}) = \frac{1}{3}$$

[4] For an observer "outside" the case, such as the jailer, the reasoning is different. For him there are two pieces of evidence: the first is that two of the three prisoners will be executed. The chance of living of each prisoner is equal (1/3). The other is that prisoner b will be executed. Then he should use Dempster's rule (within the framework of evidential theory) to combine these two pieces of evidence. One assumption for him is that if he considers the two pieces of evidence as two selections then the selections are fair.

$$(m_1 \oslash m_2)(\{(c,b)\}) = (1 \times \frac{1}{3})/(\frac{1}{2}) = \frac{2}{3}$$

Then the jailer's answer should not affect a's belief, it is still 1/3.

□

The result agrees with Bayesian analysis. This is not surprising, since the proportional sum is based on the same principle as conditional probability. The proportional sum introduced in Definition 5 is quite natural, being based the insufficient reasoning principle.

By the proportional sum operation we can define corresponding rules for other evidential functions. For example, we define belief function and plausibility function as follows.

Definition 7. Let Θ be a frame of discernment. If mass functions m_1 and m_2 on Θ are combinable then we define

$$(bel_1 \oslash bel_2)(A) = \sum_{B \subseteq A} (m_1 \oslash m_2)(B)$$

where bel_1 and bel_2 are belief functions corresponding to m_1 and m_2 respectively. Similarly for plausibility function:

$$(pls_1 \oslash pls_2)(A) = 1 - (bel_1 \oslash bel_2)(\bar{A})$$

Now we consider a special case of the proportional sum. Let Θ be a frame of discernment. Let m_1 and m_2 be two combinable mass functions on Θ. Let B be a subset of Θ and suppose m_2 satisfies that $m_2(B) = 1, m_2(elsewhere) = 0$. Suppose all focal elements of m_1 are A_1,...,A_n. If all A_i, $i = 1, ..., n$ are subsets of B, then we have $A_i \cap B = A_i$, for $i = 1, ..., n$. Therefore

$$(m_1 \oslash m_2)(A_i) = \frac{\sum_{B_j \subseteq \Theta; B_j \cap B = A_i} m_1(B_j) \cdot \frac{|A_i|}{|B_j|} \cdot m_2(B)}{\sum_{B_j, C_k \subseteq \Theta; B_j \cap C_k \neq \emptyset} m_1(B_j) \cdot \frac{|B_j \cap C_k|}{|B_j|} \cdot m_2(C_K)} = m_1(A_i)$$

while

$$(m_1 \oplus m_2)(A_i) = \frac{\sum_{B_j \subseteq \Theta; B_j \cap B = A_i} m_1(B_j) \cdot m_2(B)}{\sum_{B_j, C_k \subseteq \Theta; B_j \cap C_k \neq \emptyset} m_1(B_j) \cdot m_2(C_K)} = m_1(A_i)$$

This means that, if there is no new subset generated after updating, then updating a piece of evidence with another agrees with combining them, since we have not applied the insufficient reasoning principle in updating.

Theorem 4. Suppose m_1 and m_2 are two combinable mass functions on a frame of discernment Θ. Let B be a subset of Θ and suppose m_2 satisfies that $m_2(B) = 1, m_2(elsewhere) = 0$. Suppose all focal elements of m_1 are A_1,...,A_n. If all A_i, $i = 1, ..., n$ are subsets of B, then

$$m_1 \oslash m_2 = m_1 \oplus m_2$$

From this case we can see that the proportional sum corresponds to the point of view of belief as generalized probability, while the orthogonal sum corresponds to the point of view of belief as evidence, as indicated by Halpern and Fagin in [11]. In our framework of the D-S theory there are two rules : one is for combining evidence and another is for updating evidence.

There are some other rules for updating belief. Jeffrey considers the generalization of conditional probability, i.e. a Bayesian rule which gives a way to update probabilities when *one* event occurs. Some rules based on Jeffrey's rule are proposed[19][5][12]. Shafer considers the corresponding rule, *conditional belief*, of conditional probability in the D-S theory[13]. His rule is a combining rule rather than an updating rule compared with conditional probability. An interesting result is Fagin and Halpern's rule of updating belief proposed in [6]. The idea is that probabilities of non-measurable sets can be approximated by inner and outer probabilities and, thus, conditional probabilities can be approximated by inner and outer conditional probabilities. Then based on conditional probability they define the inner and the outer conditional probability.

5 Conclusion and further work

In this paper, by applying the insufficient reasoning principle, we propose a new rule for updating evidence. As shown by Halpern and Fagin in [11], the confusion regarding the interpretation of the Dempster and Shafer theory of evidence results from "having" two viewpoints of belief which result in two different methods to deal with evidence. Our rule for updating evidence corresponds to the viewpoint of the D-S theory as a generalization of Bayesian statistics, while Dempster's rule corresponds to the viewpoint of the D-S theory as a way of representing evidence. We make this clear by the application of the insufficient reasoning principle in the D-S theory.

We show that some of the confusion regarding interpretation of Dempster's rule of combination is because it breaks the insufficient reasoning principle, while conditional probability applies the principle implicitly. We should not expect these two rules to agree on every example. Unlike Fagin and Halpern's rule for updating beliefs [6], which essentially belongs to probability theory, our rule is defined in the same frmework as is Dempster's rule. So we have two rules in D-S theory : one is for combining evidence and another is for updating evidence.

We may follow the line of Fagin and Halpern[6] to generalize Jeffrey's rule in evidential theory, based on their results that probabilities and conditional probability of non-measurable sets can be approximated by inner and outer probabilities and conditional probabilities, and an inner measure function of a probability measure function corresponds to a belief function.

Another direction is the application of the updating rule proposed in this paper together with Dempster's rule in knowledge-based systems. Baldwin[2] gives a model for handling uncertain information which is similar to the process of applying Dempster's rule to combine evidence and then applying the updating rule to update beliefs.

References

1. S. Abel. The sum-and lattice-points method based on an evidential-reasoning system applied to the vehicle guidance problem. In *Uncertainty in Artificial Intelligence 2*, pages 365–370. 1988.
2. J. F. Baldwin. Combining evidences for evidential theory. *International Journal of Intelligent Systems*, 6(6):569–616, 1991.
3. Th. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2/3), 1988.
4. P. Diaconis. Review of "A Mathematical Theory of Evidence". *Journal of the American Statistical Society*, 73(363):677–678, 1978.
5. D. Dubois and H. Prade. Updating with belief functions, ordinal conditional functions and possibility measures. In *Uncertainty in Artificial Intelligence 6*, pages 311–329. 1991.
6. R. Fagin and J. Y. Halpern. A new approach to updating belief. In *Uncertainty in Artificial Intelligence 6*, pages 365–370. 1988.
7. R. Fagin and J. Y. Halpern. Uncertainty, belief, and probability. In *11th International Joint Conference on Artificial Intelligence*, pages 1161–1167, 1989.
8. M. Gardner. *Second Scientific American Book of Mathematical Puzzles and Diversions.* Simon & Schuster, 1961.
9. J. W. Guan and D. A. Bell. *Evidence Theory and Its Applications.* Volume 1. North-Holland, 1991.
10. P. Halmos. *Measure theory.* Van Nostrand, 1950.
11. J. Y. Halpern and R. Fagin. Two views of belief: belief as generalized probability and belief as evidence. *Artificial Intelligence*, 54:275–314, 1992.
12. H. Ichihashi and H. Tanaka. Jeffrey-like rules of conditioning for the Dempster-Shafer theory of evidence. *Inter. J. Approx. Reasoning*, 3:143–156, 1989.
13. G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.
14. Shengli Shi, M. E. C. Hull, and D. A. Bell. Evidential reasoning in expert systems. In *The 1st Workshop on Fuzzy Based Expert Systems, Sofia, Bulgaria*, 1994.
15. Shengli Shi, M. E. C. Hull, and D. A. Bell. On combining and updating evidence. In *Proceedings of the 6th International Conference on Software Engineering and Knowledge Engineering (SEKE'94)*, pages 344–351, 1994.
16. Shengli Shi, M. E. C. Hull, D. A. Bell, and J. W. Guan. Combining evidence in evidence theory. In *IEEE 1st Australian and New Zealand Conference on Intelligence Infromation System, Perth, Australia*, pages 142–146, 1993.
17. Shengli Shi, M. E. C. Hull, D. A. Bell, and J. W. Guan. Generalizing Dempster-Shafer theory of evidence. In *First European Congress on Fuzzy and Intelligent Technologies*, pages 1286–1292, 1993.
18. F. Voorbraak. On the justification of Dempster's rule of combination. *Artificial Intelligence*, 48:171–187, 1991.
19. C. G. Wagner. Generalizing jeffrey conditieneralization. In *Proceddings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 331–335, 1992.

A Global Measure of Ambiguity for Classification

Z.W. Wang and S.K.M. Wong

Department of Computer Science,
University of Regina, Regina, Saskatchewan, Canada S4S 0A2
wong@cs.uregina.ca, fax:(306)585-4745

Abstract. This paper suggests a a global measure of ambiguity based on the notion of an interval structure which can be viewed as a qualitative measure of belief. It is shown that the boundary region in the rough-set model is a special case of the proposed measure. To demonstrate the usefulness of this new measure, it is being used as a criterion for selecting appropriate attributes in the construction of decision trees.

Keywords: approximate reasoning, rough-sets, interval structure, incidence calculus, ambiguity, decision trees, classification.

1 Introduction

In the design of intelligent systems, it is often necessary to deal with imprecise information. This situation may arise from incomplete knowledge, insufficient observation, or restricted reliability of devices [Kruse et al. 91; Prerau 93]. Many numerical functions such as probability [Feller 68; Fine 73; Savage 72] and belief functions [Shafer 76, 87; Smets 88, 90] have been used for representing uncertain information. There are, however, disadvantages in using numerical (quantitative) measures as the required information may not be readily available. Another problem with purely numerical representations of uncertainty is their lack of psychological meaningfulness [Pearl 88]. Recently, non-numerical (qualitative) measures such as rough sets [Pawlak 82, 93; Slowinski 92; Ziarko 92], fuzzy sets [Zadeh 65], and interval structures [Wong et al. 92; Wong and Wang 93] are used in uncertainty management. (An interesting comparison of rough sets and fuzzy sets can be found in Dubois and Prade (88).) In the qualitative methods, decision are made without reference to numerical values. Reasoning is carried out without relying on the arithmetic manipulation of precise probabilities or belief values.

Similar to quantitative approaches, it is necessary to introduce in qualitative methods various non-numerical functions to measure vagueness or uncertainty. For example, one may represent qualitative belief by an *interval structure*. In the rough-set model, due to insufficient knowledge, a measure is introduced to estimate *ambiguity* inherent in the classification rules.

In this paper, we introduce a global measure of ambiguity. This measure is a generalized notion of the *boundary region* in rough sets. The new measure is useful for estimating classification errors particularly in those situations where more than two expert classes are involved.

This paper is organized as follows. In Section 2, we introduce a non-numerical measure of uncertainty called an interval structure. Based on this structure, a measure of ambiguity is defined for those partitions with only two equivalence classes. In Section 3, we suggest a generalized measure of ambiguity for multi-classes. We show that the measure of boundary region in rough sets is a special case. An example is given in Section 4 to illustrate the usefulness of the proposed measure.

2 Basic Notions

In this section, we define and clarify some basic notions pertinent to our discussion. We will first introduce the notion of an interval structure which consists of a pair of lower and upper mappings. We then define an ambiguity measure for dichotomic partitions. Such a measure is closely related to an interval structure.

2.1 An Interval structure

To motivate the introduction of an interval structure, let us consider a finite set of *elementary events* $\Theta = \{\theta_1, \theta_2, ..., \theta_m\}$ and a finite set of *situations* $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$. Each element in the power set 2^Θ represents a proposition. With regard to a situation $\omega \in \Omega$, we assume that a proposition $A \in 2^\Theta$ is either true or false. Ideally, one would like to define a subset $i(A) \subseteq \Omega$ to indicate that A is true for all $\omega \in i(A)$, and A is false for all $\omega \notin i(A)$ [Bundy 85]. However, due to the lack of knowledge, one may not be able to specify precisely the set of situations in which a particular proposition is true. Instead, one may be able to specify the lower and upper bounds of situations for the individual propositions. In other words, one can define two mappings $\underline{f} : 2^\Theta \rightarrow 2^\Omega$ and $\overline{f} : 2^\Theta \rightarrow 2^\Omega$ to indicate the *interval* within which the truth of a proposition lies. In fact, these two mappings define the decision rules for each proposition $A \in 2^\Theta$. That is, A is *definitely true* whenever $\omega \in \underline{f}(A)$, A is *definitely false* whenever $\omega \in \Omega - \overline{f}(A)$, and A is *possibly true* whenever $\omega \in \overline{f}(A) - \underline{f}(A)$.

We now define the pair of mappings $\underline{f}$ and $\overline{f}$ formally as follows. An upper mapping is a function, $\bar{f} : 2^\Theta \rightarrow 2^\Omega$, satisfying the axioms: for all $A, B \in 2^\Theta$,

$$\begin{array}{ll}
(\bar{f}1) & \bar{f}(\emptyset) = \emptyset, \\
(\bar{f}2) & \bar{f}(\Theta) = \Omega, \\
(\bar{f}3) & \bar{f}(A \cup B) = \bar{f}(A) \cup \bar{f}(B).
\end{array}$$

It can be easily verified that axioms $(\bar{f}1)$, $(\bar{f}2)$, and $(\bar{f}3)$ imply: for all $A, B \in 2^\Theta$,

$$(\bar{f}4) \quad \bar{f}(A \cap B) \subseteq \bar{f}(A) \cap \bar{f}(B).$$

The corresponding lower mapping, $\underline{f} : 2^{\Theta} \longrightarrow 2^{\Omega}$, is defined by: for all $A, B \in 2^{\Theta}$,

$$\underline{f}(A) = \neg\bar{f}(\neg A) \tag{1}$$

which satisfies the following properties:

$$\begin{array}{ll} (\underline{f}1) & \underline{f}(\emptyset) = \emptyset, \\ (\underline{f}2) & \underline{f}(\Theta) = \Omega, \\ (\underline{f}3) & \underline{f}(A \cap B) = \underline{f}(A) \cap \underline{f}(B), \end{array}$$

and

$$(\underline{f}4) \quad \underline{f}(A) \cup \underline{f}(B) \subseteq \underline{f}(A \cup B).$$

We call such a pair of mappings $(\underline{f}, \bar{f})$ an *interval structure.*

Alternatively, one can define an interval structure by a lower mapping, $\underline{f} : 2^{\Theta} \rightarrow 2^{\Omega}$, satisfying axioms $(\underline{f}1)$, $(\underline{f}2)$, and $(\underline{f}3)$, while its corresponding upper mapping is defined by: for all $A \in 2^{\Theta}$,

$$\bar{f}(A) = \neg\underline{f}(\neg A). \tag{2}$$

Note that axioms $(\bar{f}3)$ and $(\underline{f}3)$ imply:

$$(\bar{f}3') \quad \bar{f}(\bigcup_{i=1}^{n} A_i) = \bigcup_{i=1}^{n} \bar{f}(A_i), \tag{3}$$

$$(\underline{f}3') \quad \underline{f}(\bigcap_{i=1}^{n} A_i) = \bigcap_{i=1}^{n} \underline{f}(A_i). \tag{4}$$

An interval structure can also be defined by a *basic assignment*, $j : 2^{\Theta} \rightarrow 2^{\Omega}$, satisfying the axioms: for all $A, B \in 2^{\Theta}$,

$$\begin{array}{ll} (j1) & j(\emptyset) = \emptyset, \\ (j2) & \bigcup_{A \in 2^{\Theta}} j(A) = \Omega, \\ (j3) & A \neq B \Rightarrow (j(A) \cap j(B) = \emptyset). \end{array}$$

Lemma 1 [Wong et al. 92] *Let $\underline{f}$ and $\bar{f}$ be two mappings from 2^{Θ} to 2^{Ω}. The pair $(\underline{f}, \bar{f})$ is an interval structure, if and only if there exists a basic assignment, $j : 2^{\Theta} \rightarrow 2^{\Omega}$, such that: for all $A \in 2^{\Theta}$,*

$$\underline{f}(A) = \bigcup_{B \subseteq A} j(B), \tag{5}$$

$$\bar{f}(A) = \bigcup_{B \cap A \neq \emptyset} j(B). \tag{6}$$

2.2 An Ambiguity measure

As mentioned early, if we consider Θ to be a set of elementary events, an interval structure can be viewed as a measure of ignorance which may arise from the uncertainty about the truth or falsity of a proposition. There is another kind of ignorance referred to as *ambiguity* due to the lack of a sharp boundary in defining a concept.

Fishburn (91, 92) considered ambiguity as a primitive concept without any direct reference to likelihood, subjective probability or preference. He gave the following definition of a numerical ambiguity function.

Definition 1 *An numerical ambiguity function* $\alpha : 2^{\Theta} \longrightarrow 2^{\Omega}$ *is non-negative, vanishes at the empty set, and satisfies the complementary and submodularity axioms, namely:*

$$\begin{array}{ll}(\alpha 1) & \alpha(\emptyset) = 0, \quad \alpha(A) \geq 0,\\ (\alpha 2) & \alpha(\neg A) = \alpha(A),\\ (\alpha 3) & \alpha(A \cap B) + \alpha(A \cup B) \leq \alpha(A) + \alpha(B).\end{array}$$

Note that (α1) and (α2) imply $\alpha(\Theta) = 0$. Clearly, this function is very different from the traditional measure of uncertainty modeled by subjective probability. For example, axiom (α2) is strikingly different from the axiom $P(\neg A) = 1 - P(A)$, where P is a probability function.

More recently, Wong and Wang (93) proposed a non-numerical measure of ambiguity. Let Ω denote a set of *situations* (incidences).

Definition 2 *An non-numerical ambiguity measure, or simply an ambiguity measure, is a mapping,* $a : 2^{\Theta} \longrightarrow 2^{\Omega}$, *satisfying the following axioms: for all* $A, B \in 2^{\Theta}$,

$$\begin{array}{ll}(a1) & a(\emptyset) = \emptyset,\\ (a2) & a(A) = a(\neg A),\\ (a3.1) & a(A \cap B) \cup a(A \cup B) \subseteq a(A) \cup a(B),\\ (a3.2) & a(A \cap B) \cap a(A \cup B) \subseteq a(A) \cap a(B).\end{array}$$

It is important to note that an ambiguity measure can be expressed as the *difference* between the upper and lower mappings of an interval structure.

Lemma 2 [Wong and Wang 93] *The mapping,* $a : 2^{\Theta} \longrightarrow 2^{\Omega}$, *defined by:* for all $A \in \Theta$,

$$a(A) = \bar{f}(A) - \underline{f}(A) = \bar{f}(A) \cap \neg \underline{f}(A),$$

is a non-numerical ambiguity measure, if and only if the pair $(\underline{f}, \bar{f})$ *of mappings from* 2^{Θ} *to* 2^{Ω} *is an interval structure.*

Lemma 2 clearly indicates that there exists a close relationship between an ambiguity measure and an interval structure.

3 A Global Ambiguity Measure

Note that an ambiguity measure satisfies the property:

$$(a2) \quad a(A) = a(\neg A), \text{ for all } A \in 2^{\Omega}.$$

This suggests that we may consider an ambiguity measure as a function of partitions defined by only two equivalence classes $\{A, \neg A\}$. This can be seen as follows. By Lemma 2, an ambiguity function can be expressed as: for all $A \in 2^{\Theta}$,

$$a(A) = \bar{f}(A) \cap \neg \underline{f}(A), \tag{7}$$

where $\overline{f}$ and $\underline{f}$ are the upper and lower mappings of an interval structure. By definition,

$$\bar{f}(A) = \neg \underline{f}(\neg A).$$

It immediately follows:

$$a(A) = \neg \underline{f}(\neg A) \cap \neg \underline{f}(A) = \Omega - (\underline{f}(\neg A) \cup \underline{f}(A)). \tag{8}$$

This shows that an ambiguity measure is indeed a function of *dichotomic* partitions of Θ.

In many classification problems, one usually has to deal with more than just two expert classes. Thus, it is useful to develop an ambiguity measure for an arbitrary partition.

Definition 3 *Let $\tilde{\Theta}$ denote the set of all partition of Θ Let $\underline{f} : 2^{\Theta} \longrightarrow 2^{\Omega}$ be the lower mapping of an interval structure. A* global ambiguity measure *Amb_{Ω} is a mapping from $\tilde{\Theta}$ to 2^{Ω} defined by:*

$$Amb_{\Omega}(\tilde{A}) = \neg \bigcup_{A_i \in \tilde{A}} \underline{f}(A_i), \text{ for all } \tilde{A} = [A_1, A_2, ..., A_k] \in \tilde{\Theta}.$$

Since $\overline{f}(A) = \neg \underline{f}(\neg A)$, for all $A \in \Theta$, it is easy to see that

$$Amb_{\Omega}(\tilde{A}) = \bigcap_{A_i \in \tilde{A}} \overline{f}(\neg A_i).$$

4 An Application - Selecting Attributes for Construction of Decision Trees

Assume that the knowledge we have about a given set of objects is described by the values of a set of attributes $\{C^1, C^2, ..., C^k\}$. Such a knowledge system can be represented by a table (see Table 1), in which each object in the sample set U is characterized by a row of attribute values. The entries in the last column D of the table are the expert classification.

U	C^1	C^2	C^3	D
u_1	heavy	large	tall	+
u_2	heavy	large	tall	+
u_3	heavy	small	tall	-
u_4	heavy	medium	tall	*
u_5	heavy	small	short	+
u_6	heavy	medium	short	*

Table 1. A knowledge system described by a set of attributes.

In Table 1, the expert classification can be conveniently represented by the partition:

$$E = \{e_1 = \{u_1, u_2, u_5\}, e_2 = \{u_3\}, e_3 = \{u_4, u_6\}\}$$

of the sample set $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$. On the other hand, each attribute C^i also induces a partition Ω^i of U, namely:

$$\begin{aligned}
\Omega^1 &= \{\omega_1^1 = \{u_1, u_2, u_3, u_4, u_5, u_6\}\}, \\
\Omega^2 &= \{\omega_1^2 = \{u_1, u_2\}, \omega_2^2 = \{u_4, u_6\}, \omega_3^2 = \{u_3, u_5\}\}, \\
\Omega^3 &= \{\omega_1^3 = \{u_1, u_2, u_3, u_4\}, \omega_2^3 = \{u_5, u_6\}\}.
\end{aligned}$$

Corresponding to each partition $\Omega^i = \{\omega_1^i, \omega_2^i, ..., \omega_{k_i}^i\}$, we can construct an interval structure $(\underline{f}_{\Omega^i}, \bar{f}_{\Omega^i})$ defined by: for all $A \in 2^U$,

$$\underline{f}_{\Omega^i}(A) = \bigcup_{\omega_k^i \subseteq A} \omega_k^i,$$

$$\bar{f}_{\Omega^i}(A) = \bigcup_{\omega_k^i \cap A \neq \emptyset} \omega_k^i = \neg \underline{f}(\neg A).$$

Note that the above mappings, $\underline{f}_{\Omega^i} : 2^U \rightarrow 2^U$ and $\overline{f}_{\Omega^i} : 2^U \rightarrow 2^U$ are in fact the lower and upper approximations and $\overline{f}_{\Omega^i}(A) - \underline{f}_{\Omega^i}(A)$ is the boundary region in the rough-set model [Pawlak 82, 91].

Let $\tilde{U}$ denote the set of all partitions of U. By Definition 3, we can construct a global ambiguity measure Amb_{Ω^i} for every attribute C^i: for every partition $E \in \tilde{U}$,

$$Amb_{\Omega^i}(E) = \neg \bigcup_{e \in E} \underline{f}_{\Omega^i}(e),$$

where e is an equivalence class of E.

Let Ω and Ω' be two partitions of U. If for any equivalence class $\omega \in \Omega$, there exists an equivalence class $\omega' \in \Omega'$ such that $\omega \subseteq \omega'$, we say Ω is *finer* than Ω' or Ω' is *coarse* than Ω, written $\Omega \preceq \Omega'$.

Theorem 1 *Given a partition $E \in \tilde{U}$, for any two partitions $\Omega, \Omega' \in \tilde{U}$,*

$$\Omega \preceq \Omega' \Longrightarrow Amb_{\Omega}(E) \subseteq Amb_{\Omega'}(E).$$

Proof: Let ω, ω' denote the equivalence classes of Ω, Ω' respectively. By definition,

$$\underline{f}_{\Omega}(e) = \bigcup_{\omega \subseteq e} \omega,$$

$$\underline{f}_{\Omega'}(e) = \bigcup_{\omega' \subseteq e} \omega'.$$

For every object $u \in \bigcup_{\omega' \subseteq e} \omega'$, there exists an equivalence class $\omega' \subseteq E$ such that $u \in \omega'$. If Ω is finer than Ω', then every $\omega' \in \Omega'$ is the union of one or more ω's in Ω. Hence, there exists an equivalence class $\omega \in \Omega$ such that $u \in \omega \subseteq \omega' \subseteq e$. Therefore, $u \in \bigcup_{\omega \subseteq e} \omega$. That is,

$$\bigcup_{\omega' \subseteq e} \omega' \subseteq \bigcup_{\omega \subseteq e} \omega.$$

Thus,

$$\bigcup_{e \in E} (\bigcup_{\omega' \subseteq e} \omega') \subseteq \bigcup_{e \in E} (\bigcup_{\omega \subseteq e} \omega).$$

It follows:

$$\neg \bigcup_{e \in E} (\bigcup_{\omega \subseteq e} \omega) \subseteq \neg \bigcup_{e \in E} (\bigcup_{\omega' \subseteq e} \omega').$$

We can therefore conclude:

$$Amb_{\Omega}(E) \subseteq Amb_{\Omega'}(E). \quad \square$$

Suppose our objective here is to construct a decision tree based on the knowledge system given in Table 1. We can select an appropriate set of attributes to classify the objects in U by the criterion described below. By Theorem 1, for a fixed expert classification E, we can rank the partitions Ω^i's induced by the individual attributes C^i's by the following ranking function:

$$r(\Omega) = |Amb_E(\Omega)|,$$

where $|\cdots|$ denotes the cardinality of a set. First, the attribute C^{min} with the minimum value $r(\Omega^{min})$ is selected to classify the objects in U. It can be easily verified that in this example, $C^{min} = C^2$. Using the remaining attributes, the procedure is applied recursively to those subsets U_i's containing a mixture of objects labeled by different expert classes. As shown in Figure 1, the same procedure is applied to U_3 using attribute C^3. The procedure is being terminated if either

1. No U_i contains a mixture of objects with different expert classes, or
2. No attribute remains for classification.

Figure 1 shows the decision tree obtained by such a criterion for attribute selection.

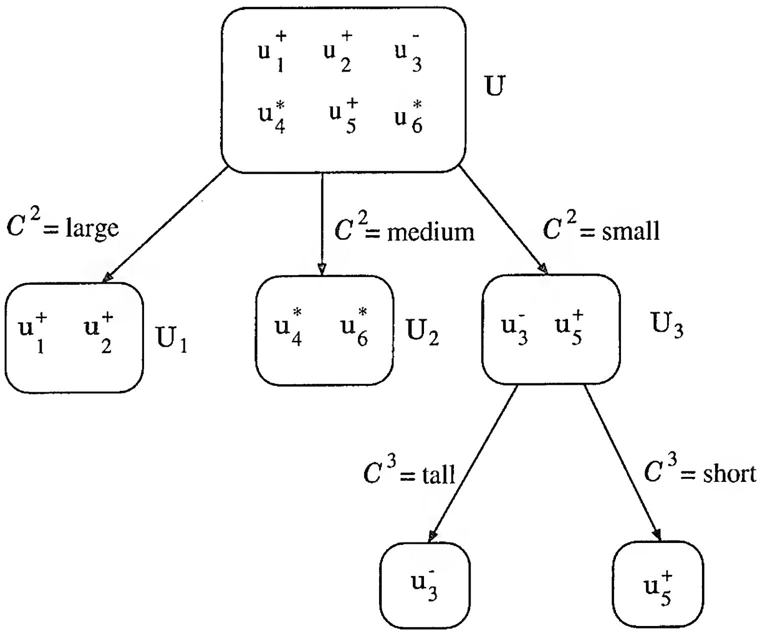


Figure 1: A decision tree obtained by the proposed attribute selection criterion.

5 Conclusion

We have proposed a global measure of ambiguity using the notion of an interval structure, which can be viewed as a qualitative measure of belief. In particular, it is shown that the boundary region in the rough-set model is a special case of the global measure for dichotomic partitions.

To demonstrate the usefulness of this preliminary result, we show that the new measure can be used to develop a criterion for selecting appropriate attributes in the construction of decision trees.

References

Bundy, A., (1985), "Incidence Calculus: a Mechanism for Probabilistic Reasoning", *Journal of Automated Reasoning*, **1**, 263-283.

Dubois, D. and Prade, H., (1988), "Rough Fuzzy Sets and Fuzzy Rough Sets", *Internal Conference on Fuzzy Sets in Informatics, Moscow, September*, 20-30.

Feller, W., (1968), *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons.

Fine, T., (1973), *Theories of Probability.* Academic Press, New York

Fishburn, P.C., (1991), "On the Theory of Ambiguity", *Information and Management Sciences*, **2**, 1-16.

Fishburn, P.C., (1992), "The Axioms and Algebra of Ambiguity", private communication.

Kruse, R., Schwecke, E., and Heinsohn, J., (1991), *Uncertainty and Vagueness in Knowledge Based Systems.* Springer-Verlag, Berlin.

Pawlak, Z., (1982), "Rough Sets", *Internal Journal of Information and Computer Sciences*, **11**, 341-356.

Pawlak, Z., (1991), *Rough Sets: Theoretical Aspects of Reasoning About Data.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Pearl, J., (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers.

Prerau, D.S., (1990), *Developing and Managing Expert Systems: Proven Techniques for Business and Industry.* Addison-Wesley Publishing Company, Inc.

Savage, L.J., (1972), *The Foundations of Statistics*, Dover, New York.

Slowinski R. (ed.), (1992), *Intelligent Decision Support Handbook of Applications and Advances of the Rough Sets Theory.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Shafer, G., (1976), *A Mathematical Theory of Evidence.* Princeton University Press, Princeton.

Shafer, G., (1987), "Belief Functions and Possibility Measures", *Analysis of Fuzzy Information, Vol. 1: Mathematics and Logic*, Bezdek, J.C. Ed. Boca Raton, FL: CRC Press, 51-84.

Smets, P., (1988), "Belief Functions (with Discussion)" in *Non-Standard Logics for Automated Reasoning.* Smets, P., Mamdani, A., Dubois, D., and Prade, H., Eds. New York: Academic, 282-285.

Smets, P., (1990), "The Combination of Evidence in the Transferable Belief Model," *IEEE Trans. Pattern Anal. Machine Intell.*, **12**, 447-458.

Wong, S.K.M., Wang, L.S., and Yao, Y.Y., (1992), "Interval Structure: a Framework for Representing Uncertain Information", *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, 336-343.

Wong, S.K.M. and Wang, Z.W., (1993), "Qualitative Measures of Ambiguity", *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, 443-450

Zadeh, L.A., (1965), "Fuzzy Sets", *Information and Control*, **8**, 338-353.

Ziarko, W., (1992), "Variable Precision Rough Set Model", *J. of Computer and System Sciences*, **46(1)**.

Meta-Level Control of Approximate Reasoning: A Decision Theoretic Approach

Shlomo Zilberstein

Computer Science Department, University of Massachusetts, Amherst, MA 01003

Abstract. This paper describes a novel methodology for meta-level control of approximate reasoning. We show that approximate reasoning performed by anytime algorithms offers a simple means by which an intelligent system can trade-off decision quality for deliberation cost. The model exploits probabilistic knowledge about the environment and about the performance of each component in order to optimally manage computational resources. An off-line knowledge compilation technique and a run-time monitoring process guarantee that the system's performance is maximized. The paper concludes with a summary of two applications.

1 Approximate Reasoning in Intelligent Systems

Approximate reasoning techniques, such as abstraction, variable precision logic, and limited horizon search, play an increasing role in intelligent systems. The need to manipulate approximate information stems from various reasons such as an imprecise model of the environment, the presence of stochastic events, limited computational resources, and noisy sensing devices. As a result, complex intelligent systems face a new control problem related to the management of precision.

When a system is composed of a number of modules that produce approximate results, important methodological questions arise regarding the management of uncertainty and precision. How can the performance of the approximate components be described? How does the output quality of a module depend on the precision of the input it receives? How should the execution of a composite system be managed so as to maximize its overall performance? And most importantly, what design methodologies simplify the task of the programmer developing such systems?

We have developed an efficient model that answers these questions. In this model, approximate reasoning is used as a valuable mechanism to trade off decision quality for deliberation costs. This mechanism allows an intelligent agent to control the level of precision of each component and maximize the achievement of its top-level goals.

The basic constructs of our model are anytime algorithms [1, 3] that form a special type of approximate reasoning. They are characterized by the gradual improvement of quality of results as a function of time. Anytime algorithms offer a simple means by which a system can trade-off decision quality for deliberation costs. In addition, the model includes a novel technique to control anytime algorithms using an adaptive, decision-theoretic approach. Efficient control of computational resources is performed by two major components: an off-line knowledge compilation process and a run-time monitoring process. By mechanizing the control of precision, we take an important step towards the widespread use of approximate reasoning.

The rest of the paper outlines our methodology and its application. Section 2 describes the notion of an anytime algorithm and its attractive properties as an approximate reasoning technique. Section 3 describes a compilation technique to compose anytime algorithms. Sections 4 describes the run-time control mechanisms. In Section 5, we briefly describe two applications. Finally, Section 6 summarizes the benefits of our approach and discusses some directions for further work.

2 Anytime Algorithms

Anytime algorithms are algorithms whose quality of results improves gradually as computation time increases. They offer a tradeoff between resource consumption and output quality. Many existing programming techniques produce useful anytime algorithms. Examples include iterative deepening search, variable precision logic, and randomized techniques such as Monte Carlo algorithms or fingerprinting. For a survey of anytime programming techniques see [8].

Various metrics can be used to measure the quality of a result produced by an anytime algorithm. From a pragmatic point of view, it may seem useful to define a *single* type of quality measure to be applied to all anytime algorithms. But in practice, different types of anytime algorithms approach the exact result in different ways. The following metrics have been proved useful in anytime algorithm construction: certainty – reflecting the degree of certainty that the result is correct, accuracy – reflecting the distance between the approximate result and the exact answer, and specificity – reflecting the level of detail of the result.

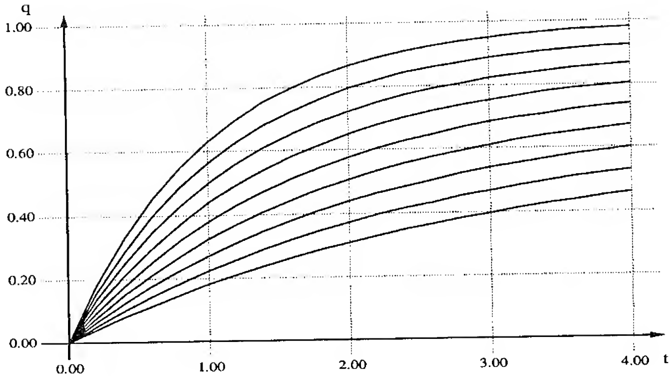


Fig. 1. Graphical representation of a CPP

2.1 Conditional Performance Profiles

To allow for efficient meta-level control of anytime algorithms, we describe their behavior by *conditional performance profiles* (CPP) [7]. A conditional performance profile captures the dependency of output quality on time allocation as well as on input quality. In [8], the reader can find a detailed discussion of various types of conditional performance profiles and their representation. To simplify the discussion of compilation, we will refer only to the *expected* CPP that maps computation time and input quality to the expected output quality.

Definition 1 *The* ***conditional performance profile (CPP)****, of an algorithm $\mathcal{A}$ is a function*

$$CPP_{\mathcal{A}} \; : \; \mathcal{Q}_{in} \times \mathcal{R}^{+} \rightarrow \mathcal{Q}_{out}$$

that maps input quality and computation time to the expected quality of the results.

Figure 1 shows a typical CPP. Each curve represents the expected output quality as a function of time for a *given* input quality.

2.2 Interruptible and Contract Algorithms

We distinguish between interruptible and contract algorithms. An interruptible algorithm is an anytime algorithm that can be interrupted at any time. A contract algorithm offers a similar tradeoff between computation time and quality of results, but the total execution time must be known in advance. If interrupted at any point before the termination of the contract time, it may yield no useful results. In many applications, interruptible algorithms are more desirable, but they are also more complicated to construct. In [6] we show that a simple, general construction can produce an interruptible version for any given contract algorithm with only a small, constant penalty. This theorem allows us to concentrate on the construction of contract algorithms for complex decision-making tasks.

2.3 A Library of Anytime Algorithms

Programming with anytime algorithms requires access to their performance profiles. For this purpose, we developed the notion of the *anytime library.* The library stores not only the performance profiles of elementary anytime algorithms, but also the results of the compilation process. The construction of a package of anytime algorithms, accompanied by a library of performance profiles, is an important first step toward the integration of approximate computation with standard software engineering techniques. Behind the anytime library concept lies the vision of a wide-spread use of standard anytime algorithms for essentially every basic computational problem from sorting and searching to complex reasoning tasks. The package of anytime algorithms supplements the compilation and monitoring techniques with flexible building blocks that simplify the development of complex systems. In current work on the development of an anytime library, we are studying such issues as machine independent representation of performance profiles, standard interface operations, and library maintenance tools.

3 Composition of Anytime Algorithms

Modularity is widely recognized as an important issue in system design and implementation. However, the use of anytime algorithms as the components of a system presents a special type of scheduling problem. The question is how much time to allocate to each component in order to maximize the output quality of the complete system. We refer to this problem as the anytime algorithm *composition problem.*

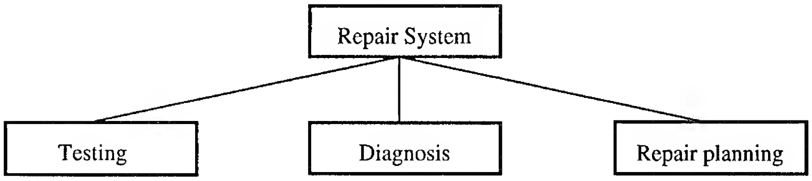


Fig. 2. A composite system for automatic diagnosis and repair

Consider for example an automated diagnosis and repair system whose structure is shown in Figure 2. The system is composed of three anytime modules that perform testing, diagnosis and repair planning. Given the conditional performance profiles of the components and a certain time allocation, the composition problem is to determine the amount of time to be allocated to each component so as to maximize the overall quality.

Solving the composition problem is important because it introduces a new kind of modularity into intelligent system development by allowing for separation between the development of the performance components and the optimization of their performance. In addition, by mechanizing the composition of anytime algorithms, we simplify the programming task.

3.1 The Compilation Problem

Given a system composed of anytime algorithms, the compilation process is designed to: (a) determine the optimal performance profile of the complete system; and (b) insert into the composite module the necessary code to achieve that performance. The precise process definition depends on various factors such as the structure of the composite program, the type of performance profiles and their representation, and the type of elementary anytime algorithms used. Depending on these factors, different types of compilation and monitoring strategies are needed. To simplify the discussion in this paper, we will consider only the problem of producing contract algorithms when the conditional performance profiles of the components are given. The reader can find a broader analysis of compilation and monitoring in [8].

Let $\mathcal{F}$ be a set of anytime functions. Assume that all function parameters are passed by value and that functions have no side-effects (as in pure functional programming). Let $\mathcal{I}$ be a set of input variables. Then, the notion of a composite expression is defined as follows:

Definition 2 *A* ***composite expression*** *over $\mathcal{F}$ with input $\mathcal{I}$ is:*

1. *An expression $f(i_1, ..., i_n)$ where $f \in \mathcal{F}$ is a function of n arguments and $i_1, ..., i_n \in \mathcal{I}$.*
2. *An expression $f(g_1, ..., g_n)$ where $f \in \mathcal{F}$ is a function of n arguments and each g_i is a composite expression or an input variable.*

For example, the expression $A(B(x), C(D(y)))$ is a composite expression over $\{A, B, C, D\}$ with input $\{x, y\}$. Suppose that each function in $\mathcal{F}$ has a conditional performance profile associated with it that specifies the quality of its output as a function of time allocation and the qualities of its inputs. Given a composite expression of size n, the compiler needs to determine the mapping, $\mathcal{T} : t \rightarrow (t_1, ..., t_n)$, that specifies the optimal time allocation to the components for any given amount of time.

3.2 Global Compilation is Hard

Global compilation of composite expressions (GCCE) refers to solving the compilation problem as a global optimization problem. In [8], we prove the following result:

Theorem 3 *The GCCE problem is NP-complete in the strong sense.*

The proof is based on a reduction from the PARTIALLY ORDERED KNAPSACK problem. This result has led us to search for efficient compilation techniques that can be applied to large programs.

3.3 Local Compilation

Local compilation is the process of finding the best performance profile of a module based on the performance profiles of its *immediate* components. If those components are not elementary anytime algorithms, then their performance profiles are determined using local compilation. Local compilation replaces the global optimization problem with a set of simpler, local optimization problems and thus reduces the complexity of the whole problem. Unfortunately, local compilation cannot be applied to every composite expression. If the expression has repeated subexpressions, then computation time should be allocated only once to evaluate all identical copies. However, the following three assumptions result in an efficient local compilation process that is also optimal [8]:

1. **The tree-structured assumption** – the input composite expression has no repeated subexpressions (thus it can be represented as a directed tree).
2. **The input-monotonicity assumption** – the output quality of each module increases when the quality of the input improves.
3. **The bounded-degree assumption** – the number of inputs to each module is bounded by a constant, b.

The first assumption is needed so that local compilation can be applied. The second assumption is needed to guarantee the optimality of the resulting performance profile. And the third assumption is needed to guarantee the efficiency of local compilation. Using an efficient tabular representation of performance profiles, we can perform local compilation in constant time and reduce the overall complexity of compilation to be linear in the size of the program. We have also developed a number of *approximate* local compilation techniques that work efficiently on DAGs and on a variety of additional programming constructs such as conditional statements and loops.

4 Meta-Level Control

Monitoring plays a central role in anytime computation. We have examined the monitoring problem in two types of domains. One type is characterized by the predictability of utility change over time. High predictability of utility allows an efficient use of contract algorithms modified by various strategies for contract adjustment. The second type of domains is characterized by rapid change and a high level of uncertainty. In such domains, scheduling interruptible algorithms based on the value of computation criterion becomes essential. Due to limited space, we discuss here only the control of contract algorithms.

Suppose that a system composed of anytime algorithms is compiled into a contract algorithm, $\mathcal{A}$. The conditional performance profile of the system is $Q_{\mathcal{A}}(q,t)$ where q is the input quality and t is the time allocation. Assume that $Q_{\mathcal{A}}(q,t)$ represents, in the general case, a probability distribution. When a discrete representation is used, $Q_{\mathcal{A}}(q,t)[q_i]$ denotes the probability of output quality q_i.

Let S_0 be the current state of the domain, let S_t represent the state of the domain at time t, and let q_t represent the quality of the result of the contract anytime algorithm at time t. $U_{\mathcal{A}}(S,t,q)$ represents the utility of a result of quality q in state S at time t. This utility function is given as part of the problem description. The purpose of the monitor is to maximize the expected utility of the result, that is, to find t for which $U_{\mathcal{A}}(S_t,t,q_t)$ is maximal.

The first step is to calculate the initial contract time. Due to the uncertainty concerning the quality of the result of the algorithm, the expected utility of the result at time t is represented by:

$$U'_{\mathcal{A}}(S_t,t) = \sum_i Q_{\mathcal{A}}(q,t)[q_i]U_{\mathcal{A}}(S_t,t,q_i) \tag{1}$$

The probability distribution of future output quality is provided by the performance profile of the algorithm. Hence, an initial contract time, t_c, can be determined before the system is activated by solving the following equation:

$$t_c = arg \max_t \{U'_{\mathcal{A}}(S_t, t)\} \tag{2}$$

One monitoring approach is to allocate the initial contract time to the components based on the compiled performance profile of the system. This approach can be improved by revising the initial allocation based on the *actual* change in the domain and the *actual* quality of partial results.

In some cases, it is possible to separate the value of the results from the time used to generate them. In such cases, one can express the comprehensive utility function, $U_{\mathcal{A}}(S, t, q)$, as the difference between two functions:

$$U_{\mathcal{A}}(S_t, t, q) = V_{\mathcal{A}}(S_0, q) - Cost(S_0, t) \tag{3}$$

where $V_{\mathcal{A}}(S, q)$ is the value of a result of quality q in a particular state S (termed *intrinsic utility* [5]) and $Cost(S, t)$ is the cost of t time units provided that the current state is S. Similar to the expected utility, the expected intrinsic utility for any allocation of time can be calculated using the performance profile of the algorithm:

$$V'_{\mathcal{A}}(S, t) = \sum_i Q_{\mathcal{A}}(q, t)[q_i] V_{\mathcal{A}}(S, q_i) \tag{4}$$

Finally, the initial contract time can be determined by solving the following equation:

$$t_c = arg \max_t \{V'_{\mathcal{A}}(S_0, t) - Cost(S_0, t)\} \tag{5}$$

Once an initial contract time is determined, several monitoring policies can be applied. In particular, we have studied two strategies for contract adjustment. The first strategy re-allocates residual time among the remaining modules once the result of a module becomes available. The second strategy adjusts the original contract time. Both of these methods are activated after the termination of each elementary component. They consider the output of that component as an input to a smaller residual system composed of the remaining anytime algorithms. At that point, a better contract time can be determined that takes into account the actual quality of the intermediate results generated so far. The higher the level of domain uncertainty, the more beneficial is the use of contract adjustments.

5 Applications

The advantages of compilation and monitoring of anytime algorithms have been demonstrated in several domains. In this section we summarize two applications.

5.1 Mobile Robot Navigation

One of the fundamental problems facing any autonomous mobile robot is the capability to plan its own motion using noisy sensory data. We have developed a simulated robot navigation system by composing two anytime modules [9]. The first module, a vision algorithm, creates a local domain description whose quality reflects the probability of correctly identifying each basic position. Each position can be free or blocked by an obstacle. The second module, a hierarchical planning algorithm, creates a path between the current position and the goal position. The quality of a plan reflects the ratio between the shortest path and the path that the robot generates when guided by the plan.

Anytime hierarchical planning is based on performing coarse-to-fine search that allows the algorithm to find quickly a low quality plan and then repeatedly refine it by replanning a segment of the plan in more detail. Hierarchical planning is complemented by an execution architecture that can take advantage of abstract plans. The execution architecture uses plans as advice to direct the base level execution mechanism. In practice, uncertainty makes it hard to use plans except as a guidance mechanism.

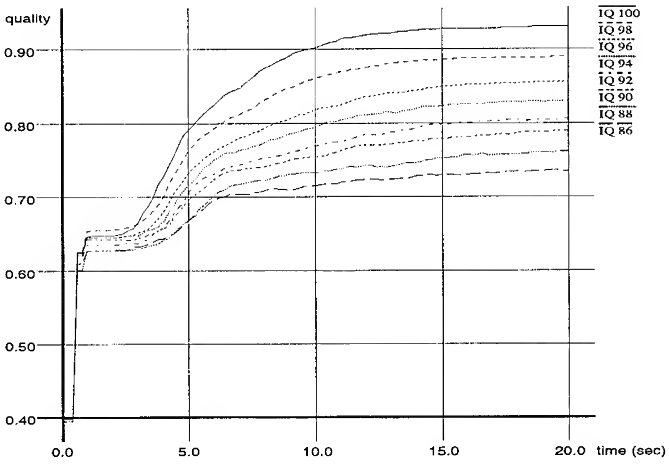


Fig. 3. The CPP of the anytime planner

The conditional performance profile of the hierarchical planner is shown in Figure 3. Each curve shows the expected plan quality as a function of run-time for a particular quality of visual data. An active monitoring scheme has also been developed to use the compiled performance profile of the system and the time-dependent utility function of the robot in order to allocate time to vision and planning so as to maximize overall utility.

An interesting result of this experiment was the fact that the anytime abstract planning algorithm produced high quality plans (approx. 10% longer than the optimal path) with time allocation that was much shorter (approx. 30%) than the total run-time of a standard search algorithm (A*). This fact shows that the flexibility of anytime algorithms does not necessarily require a compromise in overall performance.

5.2 Model-Based Diagnosis

Model-based diagnostic methods identify defective components in a system by a series of tests and probes. The goal is to locate the defective components using a small number of tests. The General Diagnostic Engine [2] (GDE) is a basic method for model-based diagnostic reasoning. In GDE, observations and a model of a system are used in order to derive *conflicts* (a conflict is a set of components of which at least one is defective). These conflicts are transformed to *diagnoses* (a diagnosis is a set of defective components that might explain the erroneous behavior of the system). The process of observation, conflict generation, transformation to diagnoses, and generation of probe advice is repeated until the defective components are identified. GDE has a high computational complexity – $O(2^n)$, where n is the number of components. As a result, its applicability is limited to small-scale applications. To overcome this difficulty, Bakker and Bourseau have developed a method, called Pragmatic Diagnostic Engine (PDE), whose computational complexity is $O(n^2)$. PDE is similar to GDE, except for omitting the stage of generating all diagnoses before determining the best measurement-point. Probe advice is given on the basis of the most relevant conflicts, called *obvious* and *semi-obvious* conflicts (an obvious (semi-obvious) conflict is a conflict that is computed using no more than one (two) observed outputs).

Pos [4] has applied our compilation technique to implement the PDE architecture. PDE can be viewed as a composition of two anytime modules. In the first module, a subset of all conflicts is determined. Pos implemented this module by a contract form of breadth-first search. The second module consists of a repeated loop that determines which measurement should be taken next, takes that measurement, and assimilates the new information into the current set of conflicts. Two versions of the diagnostic system have been implemented: one by constructing a contract algorithm and the other by making the contract system interruptible using our reduction technique. The actual slow down factor of the interruptible system was approximately 2, much better than the worst case theoretical ratio of 4.

6 Conclusion

We presented a decision-theoretic model for meta-level control of anytime algorithms. It offers both a methodological and a practical contribution to the field of real-time deliberation in intelligent systems. The main aspects of this contribution include: (1) simplifying the design and implementation of complex intelligent systems by separating the design of the performance components from the optimization of performance; (2) mechanizing the composition process and the monitoring process; and (3) constructing machine independent intelligent systems that can automatically adjust resource allocation to yield optimal performance.

The study of anytime computation is a promising and growing field in artificial intelligence. Some of the primary research directions in this field include: extending the scope of compilation by studying additional programming structures, producing a large library of reusable anytime algorithms, and developing additional, larger applications that demonstrate the benefits of the methodology.

Acknowledgements

Much of this work was done in collaboration with my graduate advisor, Stuart Russell, when I was a student at U.C. Berkeley. Tom Dean inspired my initial interest in anytime algorithms and has continued to provide important insights and comments. Current support for this project is provided by a Faculty Research Grant from the University of Massachusetts.

References

1. Dean, T. L., Boddy, M.: An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 49–54, Minneapolis, Minnesota, 1988.
2. de Kleer, J., Williams, B. C.: Diagnosing multiple faults. *Artificial Intelligence* **32**:97–130, 1987.
3. Horvitz, E. J., Breese, J. S.: Ideal partition of resources for metareasoning. Technical Report KSL-90-26, Stanford Knowledge Systems Laboratory, Stanford, California, 1990.
4. Pos, A.: *Time-Constrained Model-Based Diagnosis.* Master Thesis, Department of Computer Science, University of Twente, The Netherlands, 1993.
5. Russell, S. J., Wefald, E. H.: Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, R.J. Brachman *et al.* (eds.), San Mateo, California: Morgan Kaufmann, 1989.
6. Russell, S. J., Zilberstein, S.: Composing real-time systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 212–217, Sydney, Australia, 1991.
7. Zilberstein, S., Russell, S. J.: Efficient resource-bounded reasoning in AT-RALPH. In *Proceedings of the First International Conference on AI Planning Systems*, pp. 260–266, College Park, Maryland, 1992.
8. Zilberstein, S.: *Operational Rationality through Compilation of Anytime Algorithms.* Ph.D. dissertation, Department of Computer Science, University of California at Berkeley, 1993.
9. Zilberstein, S., Russell, S. J.: Anytime sensing, planning and action: A practical model for robot control. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1402–1407, Chambery, France, 1993.

Formal Models of Selection in Genetic Algorithms

A. Giordana F. Neri L. Saitta

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 TORINO (Italy)
{attilio, neri, saitta}@di.unito.it

Abstract. In this paper three formal models of selection operators (two known from the literature and one newly porposed) for genetic algorithms, used to learn structured concepts descriptions containing small disjuncts, are presented. The evolution of a population, according to these operators, with a generation gap equal to or less than one, is investigated in order to assess their ability to let different species emerge and coexist; finally, the average asymptotic behaviour of the population is determined. Experimental results confirm both bevahiours previously reported in the literature and the prediction of the new model.

1 Introduction

In the last years genetic algorithms proved to be an appealing alternative to traditional search algorithms because of their great exploration power and their suitability to exploit massive parallelism. They have been applied, recently, to automated induction of concept descriptions from examples, both in propositional calculus [De Jong & Spears, 1991; De Jong, Spears & Gordon, 1993; Janikov, 1991, 1993; Greene & Smith, 1993; Venturini, 1993] and in first order logic [Giordana & Saitta, 1993].

In this paper we are specifically interested in learning *disjunctive* concepts. This problem may be deceptive for a genetic search, because premature convergence to local maxima may hindern small disjuncts to be discovered [Holte et al., 1989]; in fact, larger ones tend to dominate the population, resulting either in too many, overspecific partial descriptions of the target concept, or in too few, overly general and possibly inconsistent ones. An intermediate behaviour, more or less corresponding to the intended target concept, is very difficult to obtain and critically depends upon a careful tuning of the algorithm parameters.

It has been experimentally shown that multi-modality of the fitness function can be dealt with using *sharing functions* [Debb & Goldberg, 1989; Goldberg & Richardson, 1987], which allow different species (namely, subpopulations corresponding to local maxima) to coexist. The methods of sharing functions proved effective, but is computationally expensive, because each fitness evaluation has a time complexity $O(M^2)$, being M the size of the population [Debb & Goldberg, 1989]. Moreover, when learning structured concept descriptions, sharing functions have an even more serious drawback, because they require the definition of a general, meaningful distance measure between first order logic formulæ, which is an essentially unsolved problem.

As the success of the sharing function method mainly resides in its ability to maintain for a sufficiently long time diversity inside the evolving population, giving to better hypotheses more chance to be generated through genetic recombination, the relevance of the role played by the selection process becomes apparent. Hence, we concentrate, in this paper, on the analysis of the evolution of a population, consisting of partial descriptions of a concept, under the selection operator. More specifically, we propose a formal model of a new selection operator, named the *universal suffrage* selection, and show, through a theoretical analysis of the model, that this method is particularly well suited to the considered task and has a time complexity, for the evaluation of the fitness, only of O(M). At the same time, a formal description and analysis of both the classical selection operator and of the one based on sharing functions will be supplied, for the sake of comparison. This formal analysis provides theoretical explanations of previously observed behaviours.

The above operator, together with a new search policy, has been implemented in the system REGAL, and an extensive experimentation has been performed in several domains [Giordana & Saitta, 1993; Giordana et al., 1994].

2 Definition of the Selection Operators

In this section we first define the overall problem setting and then we precisely formalize the three considered selection procedures. Let in the following E be the set of positive training examples ($|E| = N$) of the target concept. In order to obtain experimental results comparable with those predicted by the considered model, the basic algorithm used by REGAL has been simplified as follows:

```
GA :   Initialize $A(0)$ and Evaluate $A(0)$. Let $|A(0)| = M$.
       while not done do
            Select with replacement a subset $B(t)$ from $A(t)$ according to
                 the chosen selection operator. Let $|B(t)| = R$.
            Select without replacement a subset $|B'(t)|$ of R individuals
                 from $A(t)$ to be substituted. Let $G = R/M$.
            Generate $A(t+1) = A(t) - B'(t) \cup B(t)$.
```

In the above algorithm, G is the *generation gap,* i.e., the proportion of the population renewed at each generation and the probabilities of crossover and mutation have been set equal zero, so that the population evolves only under the selection operator. As a matter of fact, two different aspects contribute to the genetic evolution: the way in which individuals are introduced in the population, and the way in which individuals are kept inside the population, once added to it. Studying selection by itself corresponds to this second aspect.

Let us now precisely define the three selection operators to be investigated. Let, in the following, φ_j be a generic individual (formula of a description language L) belonging to A(t). The population $A(t) = \left\{\varphi_1^{x_1(t)}, \dots, \varphi_m^{x_m(t)}\right\}$ is a multiset, containing m different formulas ($1 \le j \le m$), each one with a multiplicity $x_j(t)$. Let $COV(\varphi_j)$ be the subset of E covered by φ_j. Then:

$$COV(A(t)) = \bigcup_{j=1}^{m} COV(\varphi_j) \qquad \text{with} \qquad |COV(A(t))| = S$$

Classical Selection Operator (CSO)

In this model, the set B(t) is selected with replacement from A(t) by assigning to each φ_j ($1 \le j \le m$) a probability proportional to its total fitness:

$$p_j = f_j\, x_j(t) \,/ \sum_{i=1}^{m} f_i\, x_i(t) \qquad (1 \le j \le m)(2.1)$$

For the specific problem at hand, the fitness $f_j = f(\varphi_j)$ should take into account completeness, consistency and (possibly, simplicity) of the formula φ_j.

The complexity of the fitness computation at each generation is O(m), because the f's value of a formula requires a constant time (its evaluation), and the summation in (2.1) is done only once. As $m \le M$, then the complexity in the worst case is of the order O(M).

Sharing Functions Selection Operator (SFO)

Given any two individuals φ_j and φ_k, let d_{kj} be their distance, evaluated either on a genotypical or on a phenotypical basis. Let s(d) be a weighting function of this distance, such as the one proposed by Goldberg [1989].

The selection operator SFO is the same as the CSO, provided that the fitness associated to each φ_j is "shared" with its neiboroughs:

$$f_j^{(s)} = f_j \, / \sum_{k=1}^{m} s(d_{jk})\, x_k(t) \qquad (2.2)$$

Then, the probability of choosing φ_j in a single trial will begiven by (2.1), where $f_j^{(s)}$ substitutes f_j.

The evaluation of all the the p_j's requires a number of steps proportional to m(m-1), which, in the worst case, is of the order $O(M^2)$.

Universal Suffrage Selection Operator (USO)

The basic idea underlying this new selection operator is that the individuals to be mated are not chosen directly from the population, but are "elected" by the available positive examples of the target concept. Each example can "vote" for one formula that covers it. The name *Universal Suffrage* selection operator comes from the fact that the right to vote is parithetically given to all the examples.

Let us now informally describe how the selection procedure works. At each generation t, a set of $R \leq M$ of examples is randomly selected with replacement from the set of positive examples E. Then, to each selected example $\xi_k \in$ E, we associate the subset $R(\xi_k)$ of A(t), containing the formulas covering ξ_k.The set $\mathbf{R}(\xi_k)$ is considered as a roulette wheel R_k, divided into sectors, each associated to a $\varphi_j \in \mathbf{R}(\xi_k)$, whose surface is proportional to the ratio between φ_j's total fitness and the sum of the fitnesses of all the formulæ in $\mathbf{R}(\xi_k)$. For each spin of the wheel just one formula is chosen. More formally:

```
Universal Suffrage
Randomly select with replacement R examples from E
for each selected examples $\xi_k$ do
    if $R(\xi_k) \neq \emptyset$ then Spin $R_k$ and put the chosen formula in $\mathbf{B}(t)$
                              else Create a new formula $\psi$ covering $\xi_k$ by applying the
                                   seeding operator[1] and add $\psi$ to $\mathbf{B}(t)$
```

Two important points are to be noted. The first one is that only formulas covering the same examples compete among each other. In this way, examples (stochastically) vote for the best of them. Another point is that the fitness function used does not need to take into account completeness, because coverage is already considered by choosing formulas by

[1] The *seeding operator* is primarily used to create a formula covering an assigned example ξ, and its role is similar to the selection of the seed in the Star methodology [Michalski, 1983; Gemello et al., 1991; Janikow, 1993]. Seeding operator is used by REGAL to create the initial population and to extend the coverage of the population with respect to the positive learning examples during selection.

means of the selection of the positive examples. As each example has the same probability of being extracted, all of them have the chance of being represented in A(t) (and hence, covered).

In order to simplify the explanation of the new selection procedure, we assume that E $\subseteq$ COV(A(t)). The selection operator can be formally described by an $(S \times m)$ stochastic matrix **P**, whose rows correspond to the examples ξ_k in COV(A(t)), and whose columns correspond to the formulæ φ_j in A(t). Each entry p_{kj} is equal to the probability that φ_j is extracted when the corresponding roulette wheel R_k is spinned:

$$\{P_{kj}\} = p_{kj} = \frac{v_{kj}\ f_j\ x_j(t)}{\sum_{i=1}^{m} v_{ki}\ f_i\ x_i(t)} \qquad \text{where} \quad v_{kj} = \begin{cases} 1 & \text{if } \xi_k \in COV(\varphi_j) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Obviously, the p_{kj}'s are normalized over the k-th row of **P**.

In order to compute the fitness values, we have to evaluate, for each φ_j, a number of sums at most equal to the number of roulette wheels available: as there is at most one roulette for each example covered by the formulas in the population, than the number of steps performed is at most N m $\leq$ N M. Hence, the global complexity is of the order O(NM).

3 Two Competing Formulas

In this section, we will study the time evolution of an "ideal average population" A(t), defined in such a way that the number of copies of each φ_j in A(t) is equal to the mean value $x_j(t)$ of the stochastic variable $x_j(t)$. We will consider the case of A(t) containing only two individuals, namely formulas φ_1 and φ_2. This simplification does not change the nature of the problem, but allows the results to be obtained in closed form. The general case will be handled in the next section.

3.1 Generation Gap G = 1

We first consider the case of a generation gap G=1, i.e., the whole population is renewed at each time step. If G =1, then R = |**B**(t)| = M. Moreover, let $x(t) = x_t$ be the number of copies of φ_1 in A(t) and $(M - x_t)$ that of φ_2.

Classical Selection Operator. According to (2.1), the probability that φ_1 is selected in one trial is given by:

$$p_t = \frac{f_1\, x_t}{f_1\, x_t + (M - x_t)\, f_2} \tag{3.1}$$

The probability distribution of the number of copies x(t+1) of φ_1 in **B**(t), given that there are x(t) copies in A(t), will be given by a binomial distribution with M trials probability of success p_t.

The mean value of x_{t+1} will be then $\overline{x}_{t+1} = M\, p_t$. Obviously, $\overline{x}_{t+1}$ depends on x_t, which is hidden inside p_t. According to what we have said before, we will consider the mean value x_{t+1} corresponding to the ideal population in which $x_t = x_t$. Then we obtain the recurrent equation:

$$x_{t+1} = \frac{\alpha \cdot x_t}{M + (\alpha - 1) \cdot x_t} \qquad (3.2)$$

where $\alpha = f_1 / f_2$. The solution of (3.2) is the following:

$$x(t) = \frac{M \cdot x_0 \cdot \alpha^t}{M - x_0 \cdot (1 - \alpha^t)} \qquad \text{with } x_0 = x(0) \quad (3.3)$$

From (3.3) we obtain the asymptotic behaviour of x(t) in A(t):

$$\lim_{t \to \infty} x(t) = \begin{cases} M & \text{if } \alpha > 1 \\ x_0 & \text{if } \alpha = 1 \\ 0 & \text{if } \alpha < 1 \end{cases} \qquad (3.4)$$

The limits (3.4) tell us that the best individual (disjunct) will (in the average) eventually take over the whole population and kill the other one. In fact, $x(t) = 0$ (for $\alpha < 1$) and $x(t) = 1$ (for $\alpha > 1$) are stable equilibrium points.

Sharing Functions Operator.

As we have only two formulas, let us denote by d their distance ($d < \sigma$) and let $\delta = s(d)$. In this case, the probability p_t simply becomes, for φ_1 :

$$p_t = \frac{\alpha}{\alpha + \frac{M - x_t}{x_t} \cdot \frac{M\delta + (1-\delta)x_t}{M - (1-\delta)x_t}} \qquad (3.5)$$

The probability distribution of x(t+1), given x(t), is the same as for the CSO, but with the probability of success given by (3.5). Then:

$$x_{t+1} = M\, p_t(x_t) \qquad (3.6)$$

Equation (3.8) has been studied in the plane (x_t, x_{t+1}). The possible equilibrium points, which give the limit values of x_t when t tends to infinity, are those in which the graph intersects the diagonal segment O*C*, i.e., those for which $x_{t+1} = x_t$. Two intersection points are trivially given by $x_1^* = 0$ and $x_2^* = M$. But there is another solution:

$$x_3^* = M \cdot \frac{\alpha - \delta}{(1-\delta) \cdot (1+\alpha)} \qquad (3.7)$$

However, x_3^* not always exists in the interval (0,1) of our interest. In fact, in order for x_3^* to belong to (0,1), it has to be:

$$\begin{cases} \delta \le \alpha \\ \alpha\delta \le 1 \end{cases} \qquad (3.8)$$

The type of equilibrium can be determined by the sign of the second derivative. The results are summarized in Fig. 1-(a) and 1-(b), for $\alpha \ge 1$ and for $\alpha < 1$, respectively.

The results of Fig. 1 explain the existence of an asymptotic equilibrium in the population among individuals with different fitness levels, i.e., the formation of species. Conditions (3.8) can be interpreted as follows: the best individual ($\alpha > 1$) shall not kill the other only if they are sufficiently far apart ($\delta < 1/\alpha$). In other words, too close maxima are merged into a single one.

$$x_t + 1 = M\, q_1 + \frac{M \cdot q_3 \cdot \alpha \cdot x_t}{M + (\alpha - 1) \cdot x_t} \qquad (3.9)$$

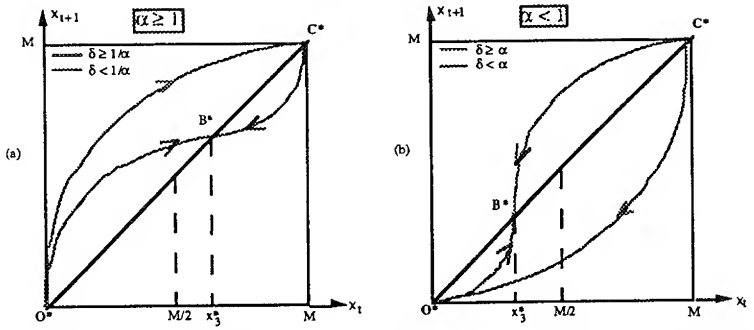


Fig. 1. Equation (3.8) in the plane (x_t, x_{t+1}). (a) Case $\alpha \geq 1$. (b) Case $\alpha < 1$. The point B*, when it exists, is a point of stable equilibrium for any value of α. For $\alpha \geq 1$ and $\delta \geq 1/\alpha$, O* is a point of unstable equilibrium and C* of stable one, whereas, for $\alpha < 1$ and $\delta \geq \alpha$, C* is unstable and O* is stable.

Universal Suffrage Operator. As we have only φ_1 and φ_2, let s_1, s_2 and s_3 be the numbers of positive examples covered only by φ_1, only by φ_2 and by both φ_1 and φ_2, respectively. We will have $S = s_1 + s_2 + s_3$. Let us define:

$$q_1 = s_1 / S \qquad q_2 = s_2 / S \qquad q_3 = s_3 / S$$

Analogously to the preceding cases, the probability distribution of x(t+1), given x(t) is a binomial with probability of success $[q_1 + q_3 p_t]$, where p_t is given by (3.1). Then:
The solution of (3.9) can be obtained in closed form:

$$x(t) = \frac{M}{\alpha - 1} \cdot \left(r_1 \cdot \frac{1 + \frac{c_2}{c_1} \cdot \left(\frac{r_2}{r_1}\right)^t}{1 + \frac{c_2}{c_1} \cdot \left(\frac{r_2}{r_1}\right)^{t-1}} - 1 \right) \qquad (3.10)$$

where:

$$r_{1,2} = 1/2 \; [\; 1+(\alpha-1)q_1 + \alpha q_1 \pm \mathrm{sqrt}([1+(\alpha-1)q_1 + \alpha q_3]^2 - 4\alpha q_3])]$$

From (3.10) it follows that:

$$\lim_{t \to \infty} \frac{x(t)}{M} = \begin{cases} \dfrac{r_1 - 1}{\alpha - 1} & \text{if } \alpha \neq 1 \\ \dfrac{q_1}{q_1 + q_2} & \text{if } \alpha = 1 \end{cases} \qquad (3.11)$$

Notice that $r_1 \geq 1$ iff $\alpha \geq 1$. Notice that, whenever a formula φ_1 is necessary to cover a part, however small, of the concept instances (i.e. $q_1 > 0$), no other formula is stochastically guaranteed to kill it in the long run, no matter how much better and how many more examples this last covers, nor how close the two formulas are. This property is fundamental for discovering small disjuncts which are essential parts of a concept description. This effect is obtained by selecting the individuals for mating through the examples. In fact, even if it may happen that φ_1 disappear from the population, it has always a chance (determined by q_1) to appear again.

Obviously, the same is true for φ_2. Then, the copies of φ_1 cannot excede, in the average, those "reserved" to φ_2, i.e., M. As a consequence, we can say that each formula maintains, inside the average population, a minimum number of copies proportional to its "private" extension, augmented by a number of copies, for which it competes with φ_2, proportional to the common extension.

It is easy to prove that a value of $G < 1$ does not change the asymptotic behaviour of the population, but only the rate of convergence to the equilibrium. In particular, $G < 1$ has the effect of *slowing down* the convergence to the equilibrium (and this is again true for all the three models).

4 Many Competing Formulas

In this section, the investigation is extended to the case in which the population consists of $m \geq 2$ formulas. Only the case $G = 1$ will be handled, because the invariance of the limit behaviour with respect to G holds also in this general case. As the method of analysis is the same as that of Section 3, we only summarize the results, omitting the rather long computations.

In the following, let $A(t) = \{\varphi_1^{x_1}, \ldots, \varphi_m^{x_m}\}$ be a multiset containing m different individuals. Obviously, the summation over the x_j equals M.

4.1 Classical Selection Operator

The probability that the new population has the multiplicities k_j for the φ_j, given the composition of A(t), is given by a multinomial distribution:

$$\Pr\{k_1, \ldots, k_m | x_1, \ldots, x_m\} = \begin{pmatrix} & M & \\ k_1 & \ldots & k_m \end{pmatrix} \cdot \prod_{i=1}^{m} p_i^{k_i} \quad (4.1)$$

By defining f(t) as the average fitness in A(t):

$$M\, f(t) = \sum_{i=1}^{m} x_i(t) \cdot f_i \quad ,$$

by computing from (4.1) the mean values $k_j = M\, p_j$ of the k_j $(1 \leq j \leq m)$, and proceding as for the case of $m = 2$, the following set of equations for the equilibrium state is obtained:

$$\frac{y_j(1-y_j)(1-\overline{\alpha_j})}{1+(\overline{\alpha_j}-1)\cdot y_j} = 0 \qquad (1 \leq j \leq m) \qquad \sum_{j=1}^{m} y_j = 1 \qquad (4.2)$$

In equations (4.2) we have defined:

$$y_j = \frac{x_j}{M} \quad \text{and} \quad \overline{\alpha_j} = \frac{f_j}{\dfrac{1}{M - x_j(t)} \displaystyle\sum_{\substack{i=1 \\ i \neq j}}^{m} x_i(t) f_i} \qquad (4.3)$$

The denominator of $\overline{\alpha_j}$ represents the average fitness of the formulas different from φ_j. Each of the first m equations (4.2) can be satisfied by $(y_j^* = 0)$ or $(y_j^* = 1)$ or $(\overline{\alpha_j} = 1)$. This last condition is verified iff $f_j = f(t)$. Moreover, $x_j(t+1) \geq x_j(t)$ iff $f_j \geq f(t)$. Then, all the formulas which have a fitness lower than f(t), decrease in average at the next generation. As a consequence, due to the normalization condition stated by the (m+1)-th equation (4.2), the number of occurrences in A(t+1) of formulas with fitness greater than f(t) increases, letting the value f(t+1) also increase. Thus, some of the formulas which had fitness greater than the average in A(t) will have a fitness lower than the new average in A(t+1), and they will start to disappear. In the limit, then, only the formula with the highest fitness value, among the φ_j, will remain in the population. By summarizing:

$$\lim_{t \to \infty} x_i(t) = \begin{cases} M & \text{if } f_j = f_{max} \\ 0 & \text{otherwise} \end{cases}$$

A special case may occur when, in a given generation A(t), a group of formulas happen to have the same fitness. In this case this fitness must be equal to the mean value. From this point on, this group of formulas remain in equilibrium (in the average), with numbers of occurrences equal to those they had in A(n), and they will kill all the others.

4.2 Sharing Functions

This case is the most computationally demanding of the three and only the results are given. In particular, an equilibrium state can be found if the following system of equations has a non trivial solution:

$$\sum_{i=1}^{m}\left(\frac{s_{ji}}{f_j}-\frac{s_{ki}}{f_k}\right)\cdot a_j = 0 \qquad (1 \le j < k \le m) \qquad (4.4)$$

System (4.4) has m (m-1)/2 homogeneous equations in the unknown a_j, which are the limit values of the x_j. Moreover, $s_{jk}= s(d_{jk})$. It is easy to prove that system (4.4) has a determinant coefficient of rank (m - 1) and it becomes determinate by adding the usual normalization condition over the a_j's. The solution of (4.4) then can be found in closed form and reduces to the solution found in Section 3 for m = 2. The conditions for the existence of the solution come from the invertibility of the coefficient matrix of the system.

4.3 Universal Suffrage

Using this selection operator, we have to extract ,with replacement, M examples from COV(A(t)). Being 1/S the probability that a single example be extracted, the probability that each ξ_k occurs h_k times is given by a multinomial distribution.

The probability distribution of the number of occurrences $x_j^{(k)}$ of each φ_j over the h_k spins of the roulette R_k is given by:

$$\Pr\{x_j^{(k)}\} = \binom{M}{x_j^{(k)}}\left(\frac{p_{kj}}{S}\right)^{x_j^{(k)}}\left[\frac{S-1}{S}+\frac{1}{S}\sum_{i\neq j}^{m} p_{kj}^{x_j^{(k)}}\right]^{M-x_j^{(k)}} \qquad (4.6)$$

Let now z_j be the total number of time φ_j will come out from the whole selection process. With G=1, z_j is equal to $x_j(t+1)$. The mean z_j value of z_j is given by:

$$z_j = x_j{}^{(k)}(t+1) = M\frac{p_{kj}(t+1)}{S} = \frac{M}{S}\,\frac{v_{kj}\, f_j\, \overline{x}_j(t)}{\sum_{i=1}^{m} v_{ki}\; f_i\; \overline{x}_i(t)} \qquad (4.7)$$

It is easy to prove that the sum over j of the x_j is equal to M. The equilibrium state is then defined by the set of equations:

$$\frac{S}{M} = \sum_{k=1}^{S} \frac{v_{kj}\, f_j}{\sum_{i=1}^{m} v_{ki}\, f_i\, \bar{x}_i} \qquad (1 \le j \le m) \qquad (4.8)$$

The solution of (4.8) can be found numerically and it can be proved that it coincides with the one found in Section 3 when m = 2. Moreover, if we define by $q_j^{(1)}$, $q_j^{(2)}$ and $q_j^{(3)}$ the proportions of examples in COV(A(t)) covered only by φ_j, not covered by φ_j and covered both by φ_j and by some other formula, respectively, it can easily be seen that:

$$M\, q_j^{(1)} \le x_j \le M\,(1 - q_j^{(2)}) \qquad (4.9)$$

The behaviour of each $x_j(t)$ satisfying the (4.9) strictly parallels the case m = 2.

5 Experimental Evaluation

A number of experiments have been performed in order to verify the theoretical predictions of the new selection operator. The results obtained in only one of the experimented domains are reported here, for the sake of brevity. More extensive experimentations are described in [Giordana, Saitta & Zini 1994].

In the chosen domain, a target concept with four disjuncts had to be identified, from a learning set of 500 positive and 600 negative instances. The first disjunct φ_1 covers 312 positive instances, the second one, φ_2, covers 167, the third one, φ_3, covers 56, and the fourth one, φ_4, covers 35. Moreover, φ_1 and φ_2 have 51 positive instances in common, whereas φ_1 and φ_4 have 19. There also exists another correct definition (discovered by REGAL) for the fourth disjunct, strongly overlapped to the second one, which covers 105 positive examples. Some time REGAL found this solution instead of the intended one.

The goal of the experiment was to check the capability of the selection algorithm to allow the formation of small disjuncts in presence of much larger ones. In particular, several experiments in different conditions have been performed in order to test: (a) the capability of the new selection operator of maintaining (on average) in the population a proportion of the four disjuncts as predicted by the solution of equations (4.8); (b) the capability of generating the disjuncts from a randomly initialized population.

In order to verify the theoretical result predicted by (4.8), REGAL has been run in 10 copies using an initial population with an identical number of copies of each one of the four disjuncts representing the target concept. The experiment has been repeated using different population sizes: 40, 80, 120, 160 and 200 individuals respectively, with a generation gap G = 1 and with crossover and mutation probabilities equal to 0. Typical results are shown in Fig. 2 and Fig. 3.

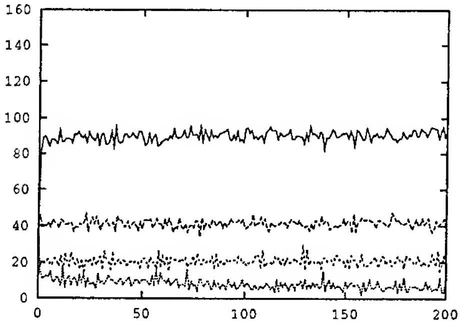


Fig. 2. Number of copies of each disjunct, averaged over the ten runs, in the case of M = 160, versus the generation number.

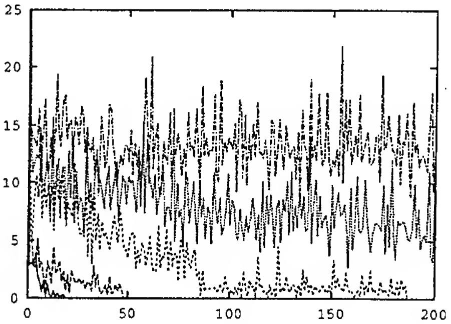


Fig. 3. Number of copies (averaged over ten runs) of the fourth disjunct (the smallest one), versus generation number. Each graph corresponds to a different value of the poulation size: M = 40, 80, 120, 160, 200.

The asymptotic values predicted by the solution of equations (4.8) for the four disjuncts are the following:

$(x_1 = 92, \sigma_1 = 6.2)$ $(x_2 = 42, \sigma_1 = 5.2)$

$(x_3 = 19, \sigma_3 = 4)$ $(x_4 = 7, \sigma_4 = 4.8)$

From Fig. 3 one can see that only for $M \geq 120$ the smallest disjunct survives at least on some of the nodes. The value predicted for 0.9 probability of not disappearing is 128, a little higher, because of the approximation done in the computations.

6 Conclusions

In this paper a new model of the selection operator, particularly well suited to be used in genetic search for learning disjunctive concept descriptions has been presented and analyzed. The model is theoretically provable superior to other existing models with respect to the ability of discovering small disjuncts. This aspect also emerged from an extensive exprimentation [Giordana & Saitta, 1993; Giordana et al., 1994].

On the other hand, the basic ideas underlaying REGAL are not so new by considering other genetic algorithms designed for the same class of tasks. The system GIL (Janikov, 1991, 1993), for instance, explicitly uses the learning instances for focusing the genetic search. Even more, the system COGIN (Greene & Smith, 1993) uses a replacement operator which is founded on the same principle as the *universal suffrage* is.

However, the substantial step ahead with respect to this similar approaches, is the development of a formal model which allows the behaviour of the system to be predicted and to set critical constants such as the size of the population to be used. Nevertheless we think it should be worth a comparison with the mentioned systems as well as with other methods (Spears, 1993) recently proposed for allowing the formation of niches and species.

Further developments of the model would include the effects of the mating policy and of the recombination operators, as well as those of a migration rate different from zero among the nodes of the network.

References

Debb K. and Goldberg D. (1989). "An Investigation of Niche and Species Formation in Genetic Function Optimization", Proc. *3rd Int. Conf. on Genetic Algorithms* (Fairfax, VA), pp. 42-50.

De Jong K.A. and Spears W.M. (1991). "Learning Concept Classification Rules Using Genetic Algorithms", Proc. *IJCAI-91* (Sidney, Australia), pp. 651-656.

De Jong, K.A. Spears W.M. and Gordon F.D. (1991). "Using genetic Algorithms for Concept Learning", *Machine Learning, 13* , 161-188.

Giordana A. and Saitta L. (1993). "REGAL: An Integrated System for Relations Using Genetic Algorithms", Proc. *2nd. International Workshop on Multistrategy Learning* (Harpers Ferry, WV), pp. 234-249.

Giordana A., L. Saitta and F. Zini (1994). "Learning Disjunctive Concepts with Genetic Algortithms". Proc. *Machine Learning Conference 94* (New Brunswick, NJ), In press.

Goldberg D.E. (1989). *Genetic Algorithms,* Addison-Wesley.

Goldberg D.E. and Richardson J. (1987). "Genetic Algorithms with Sharing for Multimodal Function Optimization", Proc. *2nd Int. Conf. on Genetic Algorithms* (Cambridge, MA), pp. 41- 49.

Greene D.P. and Smith S.F., "Competition-Based Induction of Decision Models from Examples", *Machine Learning, 13* , 229-258.

Holte R., Acker L and Porter B. (1989). "Concept Learning and the Problem of Small Disjuncts". Proc. IJCAI-89 (Detroit, MI).

Janikov C.Z. (1993). "A Knowledge Intensive Genetic Algorithm for Supervised Learning", *Machine Learning, 13* , 198-228.

Vafaie H. and De Jong K.A. (1991). "Improving the Performance of Rule Induction System Using Genetic Algorithms", Proceedings *First International Workshop on Multistrategy Learning*, (Harpers Ferry, WV), pp. 305-315.

Venturini G. (1993) "SIA: a Supervised Inductive Algorithm with Genetic Search for Learning Attribute Based Concepts", Proceedings *European Conference on Machine Learning, ECML-93* (Vienna, Austria), pp. 280-296.

Genetic Algorithms for the 0/1 Knapsack Problem

Zbigniew Michalewicz[1] and Jarosław Arabas[2]

[1] Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA

[2] Institute of Electronics Fundamentals, Warsaw University of Technology, Poland

Abstract. In this paper the utility of several constraint-handling techniques is investigated on the basis of a family of 0/1 knapsack problems. Several evolutionary algorithms are applied to this NP-hard problem. The conclusions might be applicable to many constrained combinatorial optimization problems, for which the use of evolutionary algorithm is considered.

1 Introduction

During the last twenty years genetic algorithms [6] and other evolutionary algorithms [11] have been applied to many hard problems with very good results. However, for many constrained problems the results were mixed. It seems, that (in general) there has not been any single accepted strategy to deal with constrained problems: most researchers used some ad-hoc methods for handling problem specific constraints. The reason for this phenomena might be that there is an experimental evidence [10] that incorporation of the problem specific knowledge (i.e., the problem's constraints) into the evolutionary algorithm (i.e., into its chromosomal structures and genetic operators) enhances its performance in a significant way.

The constraint-handling techniques for evolutionary algorithms can be grouped into a few categories. One way of dealing with candidates that violate the constraints is to generate potential solutions without considering the constraints and then to penalize them by decreasing the "goodness" of the evaluation function. In other words, a constrained problem is transformed to an unconstrained one by associating a penalty with all constraint violations; these penalties are included in the function evaluation. Of course, there are a variety of possible penalty functions which can be applied. Some penalty functions assign a constant as a penalty measure. Other penalty functions depend on the degree of violation: the larger violation is, the greater penalty is imposed (however, the growth of the function can be logarithmic, linear, quadratic, exponential, etc. with respect to the size of the violation). Each of these categories of penalty functions has its own disadvantages; in [4] Davis wrote:

> "If one incorporates a high penalty into the evaluation routine and the domain is one in which production of an individual violating the constraint is likely, one runs the risk of creating a genetic algorithm that

> spends most of its time evaluating illegal individuals. Further, it can happen that when a legal individual is found, it drives the others out and the population converges on it without finding better individuals, since the likely paths to other legal individuals require the production of illegal individuals as intermediate structures, and the penalties for violating the constraint make it unlikely that such intermediate structures will reproduce. If one imposes moderate penalties, the system may evolve individuals that violate the constraint but are rated better than those that do not because the rest of the evaluation function can be satisfied better by accepting the moderate constraint penalty than by avoiding it".

Also other researchers [16], [14], [17] investigated some properties of penalty functions.

Additional version of penalty approach is elimination of non-feasible solutions from the population (i.e., application of the most severe penalty: death penalty). This technique was used successfully in evolution strategies [2] for numerical optimization problems. However, such approach has its drawbacks. For some problems the probability of generating (by means of standard genetic operators) a feasible solution is relatively small and the algorithm spends a significant amount of time evaluating illegal individuals. Moreover, in this approach non-feasible solutions do not contribute to the gene-pool of any population.

Another category of constraint handling methods is based on application of special repair algorithms to "correct" any infeasible solutions so generated. Again, such repair algorithms might be computationally intensive to run and the resulting algorithm must be tailored to the particular application. Moreover, for some problems the process of correcting a solution may be as difficult as solving the original problem.

The third approach concentrates on the use of special representation mappings (decoders) which guarantee (or at least increase the probability of) the generation of a feasible solution or the use of problem-specific operators which preserve feasibility of the solutions [9]. However, decoders are frequently computationally intensive to run [4], not all constraints can be easily implemented this way, and the resulting algorithm must be tailored to the particular application.

In this paper we examine the above techniques on one particular problem: the 0/1 knapsack problem. The problem is easy to formulate, yet, the decision version of it belongs to a family of NP-complete problems. It is an interesting exercise to evaluate the advantages and disadvantages of constraint handling techniques on this particular problem with a single constraint: the conclusions might be applicable to many constrained combinatorial optimization problems.

The paper is organized in the following way. The next section introduces a version of the 0/1 knapsack problem considered in this paper and discusses the data sets used in the experiments. Section 3 presents three categories of GA-based algorithms developed for this project, and Section 4 contains the results of various experiments. The last section indicates directions for the future work.

2 The 0/1 Knapsack Problem and the Test Data

There is a variety of knapsack-type problems in which a set of entities, together with their values and sizes, is given, and it is desired to select one or more disjoint subsets so that the total of the sizes in each subset does not exceed given bounds and the total of the selected values is maximized [8]. Many of the problems in this class are NP-hard and large instances of such problems can be approach only by using heuristic algorithms. The problem studied in this paper is the 0/1 knapsack problem. The task is, for a given set of weights $W[i]$, profits $P[i]$, and capacity C, to find a binary vector $\mathbf{x} = \langle x[1], \ldots, x[n]\rangle$, such that

$\sum_{i=1}^{n} x[i] \cdot W[i] \leq C$,

and for which

$\mathcal{P}(\mathbf{x}) = \sum_{i=1}^{n} x[i] \cdot P[i]$

is maximum.

In this paper we analyse the experimental behavior of a few GA-based algorithms on several sets of randomly generated test problems. Since the difficulty of such problems is greatly affected by the correlation between profits and weights [8], we consider three randomly generated sets of data:

- *uncorrelated*:
  $W[i] :=$ (uniformly) random($[1..v]$), and $P[i] :=$ (uniformly) random($[1..v]$).
- *weakly correlated*:
  $W[i] :=$ (uniformly) random($[1..v]$), and
  $P[i] := W[i] +$ (uniformly) random($[-r..r]$), (if, for some i, $P[i] \leq 0$, such profit value is ignored and the calculations are repeated until $P[i] > 0$).
- *strongly correlated*:
  $W[i] :=$ (uniformly) random($[1..v]$), and $P[i] := W[i] + r$.

Higher correlation implies smaller value of the difference:

$\max_{i=1..n}\{P[i]/W[i]\} - \min_{i=1..n}\{P[i]/W[i]\}$;

as reported in [8], higher correlation problems have higher expected difficulty.

Data have been generated with the following parameter settings: $v = 10$ and $r = 5$. For the tests we used three data sets of each type containing $n = 100$, 250, and 500 items, respectively. Again, following a suggestion from [8], we have taken under consideration two knapsack types:

- *restrictive knapsack capacity*
  A knapsack with the capacity of $C_1 = 2v$. In this case the optimal solution contains very few items. An area, for which conditions are not fulfilled, occupies almost the whole domain.
- *average knapsack capacity*
  A knapsack with the capacity $C_2 = 0.5 \sum_{i=1}^{n} W[i]$. In this case about half of the items are in the optimal solution.

As reported in [8], further increasing the value of capacity C does not significantly increase the computation times of the classical algorithms.

3 Description of the Algorithms

We have implemented and tested three types of algorithms: algorithms based on penalty functions ($A_p[i]$, where i is the index of a particular algorithm in this class), algorithms based on repair methods ($A_r[i]$), and algorithms based on decoders ($A_d[i]$). We discuss these three categories in the following subsections.

3.1 Algorithms $A_p[i]$

In all algorithms in this category a binary string of the length n represents a solution $\mathbf{x}$ to the problem: the i-th item is selected for the knapsack iff $x[i] = 1$.

The fitness $eval(\mathbf{x})$ of each string is determined as:

$$eval(\mathbf{x}) = \sum_{i=1}^{n} x[i] \cdot P[i] - Pen(\mathbf{x}),$$

where penalty function $Pen(\mathbf{x})$ is zero for all feasible solutions $\mathbf{x}$, i.e., solutions such that $\sum_{i=1}^{n} x[i] \cdot W[i] \leq C$, and is greater than zero, otherwise.

There are many possible strategies for assigning the penalty value. In this paper we consider three cases only, where the growth of the penalty function is logarithmic, linear, and quadratic with respect to the degree of violation, respectively:

- $A_p[1]$: $Pen(\mathbf{x}) = \log_2(1 + \rho \cdot (\sum_{i=1}^{n} x[i] \cdot W[i] - C))$,
- $A_p[2]$: $Pen(\mathbf{x}) = \rho \cdot (\sum_{i=1}^{n} x[i] \cdot W[i] - C)$,
- $A_p[3]$: $Pen(\mathbf{x}) = (\rho \cdot (\sum_{i=1}^{n} x[i] \cdot W[i] - C))^2$.

In all three cases, $\rho = \max_{i=1..n}\{P[i]/W[i]\}$.

3.2 Algorithms $A_r[i]$

As in the previous category of algorithms, a binary string of the length n represents a solution to the problem $\mathbf{x}$: the i-th item is selected for the knapsack iff $x[i] = 1$.

The fitness $eval(\mathbf{x})$ of each string is determined as:

$$eval(\mathbf{x}) = \sum_{i=1}^{n} x'[i] \cdot P[i],$$

where vector $\mathbf{x}'$ is a repaired version of the original vector $\mathbf{x}$.

There are two interesting aspects here. First, we may consider different repair methods. Second, some percentage of repaired chromosomes may replace the original chromosomes in the population. Such replacement rate may vary from 0% to 100%; recently Orvosh and Davis [13] reported so-call 5% rule which states that if replacing original chromosomes with a 5% probability, the performance of the algorithm is better than if replacing with any other rate (in particular, it is better than with 'never replacing' or 'always replacing' strategies).

We have implemented and tested two different repair algorithms. Both algorithms are based on the same procedure, shown in Figure 1.

The two repair algorithms considered in this paper differ only in selection procedure **select**, which chooses an item for removal from the knapsack:

```
procedure repair (x)
begin
    knapsack-overfilled := false
    x' := x
    if $\sum_{i=1}^{n} x'[i] \cdot W[i] > C$
        then knapsack-overfilled := true
    while (knapsack-overfilled) do
    begin
        $i$ := select an item from the knapsack
        remove the selected item from the knapsack:
            i.e., $x'[i] := 0$
        if $\sum_{i=1}^{n} x'[i] \cdot W[i] \leq C$
        then knapsack-overfilled := false
    end
end
```

Fig. 1. The repair procedure

- $A_r[1]$ (random repair). The procedure **select** selects a random element from the knapsack.
- $A_r[2]$ (greedy repair). All items in the knapsack are sorted in the decreasing order of their profit to weight ratios. The procedure **select** chooses always the last item (from the list of available items) for deletion.

3.3 Algorithms $A_d[i]$

The most natural decoders for the knapsack problem are based on integer representation. In this paper we used the ordinal representation of selected items. Each chromosome is a vector of n integers; the i-th component of the vector is an integer in the range from 1 to $n - i + 1$. The ordinal representation references a list L of items; a vector is decoded by selecting appropriate item from the current list. For example, for a list of items $L = (1, 2, 3, 4, 5, 6)$, the vector $\langle 4, 3, 4, 1, 1, 1\rangle$ is decoded as the following sequence of items: 4, 3, 6 (since 6 is the 4-th element on the current list after removal of 4 and 3), 1, 2, and 5. Clearly, in this method a chromosome can be interpreted as a strategy of incorporating items into the solution. Additionally, one-point crossover applied to any two feasible parents would produce a feasible offspring. A mutation operator is defined in a similar way as for the binary representation: if the i-th gene undergoes mutation, it takes a value random value (uniform distribution) from the range $[1..n - i + 1]$. The decoding algorithm is presented in Figure 2.
The two algorithms based on decoding techniques considered in this paper differ only in the procedure **build**:

- $A_d[1]$ (random decoding). In this algorithm the procedure **build** creates a list L of items such that the order of items on the list corresponds to the order of items in the input file (which is random).

```
procedure decode (x)
begin
    build a list $L$ of items
    $i := 1$
    $WeightSum := 0$
    $ProfitSum := 0$
    while $i \leq n$ do
    begin
        $j := x[i]$
        remove the $j$-th item from the list $L$
        if $WeightSum + W[j] \leq C$ then
        begin
            $WeightSum := WeightSum + Weight[j]$
            $ProfitSum := ProfitSum + Profit[j]$
        end
        $i := i + 1$
    end
end
```

Fig. 2. The decoding procedure for the ordinal representation

- $A_d[2]$ (greedy decoding). The procedure **build** creates a list L of items in the decreasing order of their profit to weight ratios. The decoding of the vector $\mathbf{x}$ is done on the basis of the sorted sequence (there are some similarities with the $A_r[2]$ method). For example, $x[i] = 23$ is interpreted as the 23-rd item (in the decreasing order of the profit to weight ratios) on the current list L.

4 Experiments, Results and Conclusions

In all experiments the population size was constant and equal to 100. Also, probabilities of mutation and crossover were fixed: 0.05 and 0.65, respectively. We used a simple one-point crossover. As a performance measure of the algorithm we have collected the best solution found within 500 generations. It has been empirically verified that after such number of generations no improvement has been observed. The results reported in the Table 1 are mean values of the 25 experiments.[3]. Note, that the data files were unsorted (arbitrary sequences of items, not related to their $P[i]/W[i]$ ratios). The capacity types C_1 and C_2 stand for restrictive and average capacities (Section 2), respectively.

Results for the methods $A_r[1]$ and $A_r[2]$ have been obtained using the 5% repair rule. We have also examined whether the 5% rule works for the 0/1 knapsack problem (the rule was discovered during experiments on two other combinatorial problems: network design problem and graph coloring problem [13]). For the

[3] In the full version of the paper we will compare the above techniques also with the standard branch & bound method

Correl.	No. of	Cap.	method						
	items	type	$A_p[1]$	$A_p[2]$	$A_p[3]$	$A_r[1]$	$A_r[2]$	$A_d[1]$	$A_d[2]$
none	100	C_1	*	*	*	62.9	94.0	63.5	59.4
		C_2	398.1	341.3	342.6	344.6	371.3	354.7	353.3
	250	C_1	*	*	*	62.6	135.1	58.0	60.4
		C_2	919.6	837.3	825.5	842.2	894.4	867.4	857.5
	500	C_1	*	*	*	63.9	156.2	61.0	61.4
		C_2	1712.2	1570.8	1565.1	1577.4	1663.2	1602.8	1597.0
weak	100	C_1	*	*	*	39.7	51.0	38.2	38.4
		C_2	408.5	327.0	328.3	330.1	358.2	333.6	332.3
	250	C_1	*	*	*	43.7	74.0	42.7	44.7
		C_2	920.8	791.3	788.5	798.4	852.1	804.4	799.0
	500	C_1	*	*	*	44.5	93.8	43.2	44.5
		C_2	1729.0	1531.8	1532.0	1538.6	1624.8	1548.4	1547.1
strong	100	C_1	*	*	*	61.6	90.0	59.5	59.5
		C_2	741.7	564.5	564.4	566.5	577.0	576.2	576.2
	250	C_1	*	*	*	65.5	117.0	65.5	64.0
		C_2	1631.9	1339.5	1343.4	1345.8	1364.4	1366.4	1359.0
	500	C_1	*	*	*	67.5	120.0	67.1	64.1
		C_2	3051.6	2703.8	2700.8	2709.5	2748.1	2738.0	2744.0

Table 1. Results of experiments; symbol '*' means, that no valid solution has been found within given time constraints

purpose of comparison we have chosen the test data sets with the weak correlation between weights and profits. All parameters settings were fixed and the value of the repair ratio varied from 0% to 100%. We have observed no influence of the 5% rule on performance of the genetic algorithm. The results (algorithm $A_r[2]$) have been collected in the Table 2.

The main conclusions drawn from the experiments can be summarized as follows:

- Penalty functions $A_p[i]$ (for all i) do not produce feasible results on problems with restrictive knapsack capacity (C_1). This is the case for any number of items ($n = 100$, 250, and 500) and any correlation.
- Judging only from the results of the experiments on problems with average knapsack capacity (C_2), the algorithm $A_p[1]$ based on logarithmic penalty function is a clear winner: it outperforms all other techniques on all cases (uncorrelated, weakly and strongly correlated, with $n = 100$, 250, and 500 items). However, as mentioned earlier, it fails on problems with restrictive capacity.
- Judging only from the results of the experiments on problems with restrictive knapsack capacity (C_1), the repair method $A_r[2]$ (greedy repair) outperforms all other methods on all test cases.

These results are quite intuitive. In the case of restrictive knapsack capacity,

correlation	weak					
no. of items	100		250		500	
capacity type	C_1	C_2	C_1	C_2	C_1	C_2
repair ratio						
0	94.0	371.3	134.1	895.0	158.0	1649.1
5	94.0	371.7	135.1	891.4	155.8	1648.5
10	94.0	370.2	135.1	889.5	157.3	1640.3
15	94.0	368.3	135.3	895.4	156.6	1646.2
20	94.0	372.0	135.6	905.7	155.8	1643.6
25	94.0	370.2	135.1	894.0	155.8	1644.0
30	94.0	367.2	135.1	895.7	157.3	1648.0
35	94.0	370.3	136.1	896.5	156.3	1643.3
40	94.0	368.6	134.3	886.5	156.1	1648.4
45	94.0	369.0	135.3	891.5	156.6	1649.0
50	94.0	371.7	134.6	891.0	156.1	1641.5
55	94.0	371.3	135.0	895.7	157.0	1647.0
60	94.0	369.6	135.0	894.0	156.1	1645.0
65	94.0	370.0	135.1	893.2	156.6	1642.8
70	94.0	367.6	135.0	893.4	156.1	1640.9
75	94.0	367.7	135.3	895.7	157.1	1648.1
80	94.0	368.2	134.3	898.5	155.8	1648.5
85	94.0	364.7	135.6	897.4	156.1	1646.2
90	94.0	368.7	134.3	885.2	156.1	1648.9
95	94.0	371.2	135.0	890.5	155.3	1642.3
100	94.0	370.2	134.6	901.0	156.3	1646.1

Table 2. Influence of the repair ratio on the performance of the algorithm $A_r[2]$

only very small fraction of possible subsets of items constitute feasible solutions; consequently, most penalty methods would fail. This is generally the case for many other combinatorial problems: the smaller the ratio between feasible part of the search space and the whole search space, the harder it is for the penalty function methods to provide feasible results. On the other hand, the repair algorithms perform quite well. In the case of average knapsack capacity, logarithmic penalty function method (i.e., small penalties) is superior: it is interesting to note that the size of the problem does not influence the conclusions.

5 Future Work

This paper reports some preliminary results on applying several genetic algorithms for the 0/1 knapsack problem. We investigated the influence of the constraint handling method on the performance of the algorithm. This research is still during its early stages; many additional experiments are planned in the near future.

In the category of penalty functions, it might be interesting to experiment with additional formulae. All considered penalties were of the form $Pen(\mathbf{x}) = f(\mathbf{x})$, where f was logarithmic, linear, and quadratic. Another possibility would be to experiment with $Pen(\mathbf{x}) = a+f(\mathbf{x})$ for some constant a. This would provide a mimimum penalty for any infeasible vector. Further, it might be interesting to experiment with *dynamic* penalty functions, where their values depend on additional parameters, like the generation number (this was done in [12] for numerical optimization for continuous variables). Also, it seems worthwhile to experiment with *adaptive* penalty functions. After all, probabilities of applied operators might be adaptive (as in evolution strategies); some initial experiments indicate that adaptive population sizes may have some merit [1]; so the idea of adaptive penalty functions deserves some attention. In its simplest version, a penalty coefficient would be part of the solution vector and undergo all genetic (random) changes (as opposed the idea of dynamic penalty functions, where such penalty coefficient is changed on regular basis as a function of, for example, generation number).

It is also possible to experiment with many repair schemes, including other heuristics than the ratio of the profit and the weight. Also, it might be interesting to combine the penalty methods with repair algorithms: infeasible solutions for algorithms $A_p[i]$ could have been repaired into feasible ones.

In the category of decoders it is necessary to experiment with different (integer) representations (as it was done for the traveling salesman problem): adjacency representation (with alternating-edges crossover, subtour-chunks crossover, or heuristic crossover), or path representation (with the PMX, OX, and CX crossovers, or even the edge recombination crossover). It would be interesting to compare the usefulness of these representations and operators for the 0/1 knapsack problem (as it was done for the traveling salesman problem and scheduling [18]). It is quite possible, that some new crossover (based on some heuristic) would provide with the best results.

References

1. Arabas, J., Michalewicz, Z., and Mulawka, J., *GAVaPS — a Genetic algorithm with Varying Population Size*, Technical Report 001-1994, Department of Computer Science, University of North Carolina, Charlotte, 1994.
2. Bäck, T., Hoffmeister, F., and Schwefel, H.-P., *A Survey of Evolution Strategies*, in [3], pp.2–9.
3. Belew, R. and Booker, L. (Editors), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1991.
4. Davis, L., (Editor), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.
5. Forrest, S. (Editor), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1993.
6. Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

7. Khuri, S., Bäck, T., and Heitkötter, J., *The Zero/One Multiple Knapsack Problem and Genetic Algorithms*, Proceedings of the ACM Symposium of Applied Computation (SAC '94).
8. Martello, S. and Toth, P., *Knapsack Problems*, John Wiley & Sons, Chichester, UK, 1990.
9. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, 1992.
10. Michalewicz, Z., *A Hierarchy of Evolution Programs: An Experimental Study*, Evolutionary Computation, Vol.1, No.1, 1993, pp.51–76.
11. Michalewicz, Z. (Ed.), Statistics & Computing, special issue on evolutionary computation, 1994.
12. Michalewicz, Z., and Attia, N., *Evolutionary Optimization of Constrained Problems*, 3rd Annual Conference on Evolutionary Programming, February 24–26, 1994, San Diego.
13. Orvosh, D. and Davis, L., *Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints*, in [5], p.650.
14. Richardson, J.T., Palmer, M.R., Liepins, G., and Hilliard, M., *Some Guidelines for Genetic Algorithms with Penalty Functions*, in [15], pp.191–197.
15. Schaffer, J., (Editor), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, CA, 1989.
16. Siedlecki, W. and Sklanski, J., *Constrained Genetic Optimization via Dynamic Reward–Penalty Balancing and Its Use in Pattern Recognition*, in [15], pp.141–150.
17. Smith, A. and Tate, D., *Genetic Optimization Using A Penalty Function*, in [5], pp.499–503.
18. Starkweather, T., McDaniel, S., Mathias, K., Whitley, C., and Whitley, D., *A Comparison of Genetic Sequencing Operators*, in [3], pp.69–76.

On the Weakening of Fuzzy Relational Queries

[1] Troels Andreasen, [2] Olivier Pivert

[1] Roskilde University
PB 260 DK-4000 ROSKILDE
DENMARK

[2] IRISA-ENSSAT
BP 447 22305 LANNION Cédex
FRANCE

e-mail : troels@ruc.dk, pivert@enssat.fr

Abstract. In this paper, we present a cooperative approach intended to avoid empty answers to conjunctive fuzzy relational queries. We propose a weakening mechanism generating more tolerant queries. This consists in relaxing one or several fuzzy predicates with the help of a fuzzy linguistic modifier which is used to build a lattice of weakened queries expressing a semantic distance between these queries. Selectivity is then used as a heuristic to guide through the lattice, in order to find the query with a non-empty answer which is as close as possible to the initial one.

1 Introduction

In the database domain, a promising research field concerns the introduction of fuzzy filtering capabilities in querying systems in order to make them more flexible. In this paper, we consider databases where the data is usual i.e. totally known or unknown. The main objective of a flexible system is to discriminate answers by performing a qualitative distinction of their satisfaction with respect to the query submitted by the user. Several approaches can be taken, and some of them have been proposed and implemented in research prototypes. It has been shown in [1] that these approaches only constitute particular cases of the one which is based on the fuzzy set theory.

Traditional query processing systems only accept precisely specified queries and only provide exact answers, thus requiring users to fully understand the problem domain and the database schema, and still often returning useless or even null information when exact answers do not cover what is intended or are not available. To remedy the latter restriction, extending the classical notion of query answering to cooperative answering has been explored [6, 7, 11]. Flexible query processing systems attack the former restriction by allowing imprecise specification of queries, and furthermore reduce the risk of obtaining a null answer to a certain extent, as the imprecision fringe in the formulation of the user queries increases the chances that an element satisfies the filter. Nevertheless, the case can occur where none of the elements satisfy the criterion formulated by the user. In such a situation, two kinds of solution are conceivable : i) giving the user information explaining why there is no answer, ii) weakening the query i.e. making it less restrictive in order to obtain a non empty answer. As noted by Motro [12], two causes exist in general for empty answers : either the user query shows a "semantic defect" or the only reason resides in the current database state. In case of a semantic defect in the query, an explanation can then enlighten the user. This usually necessitates additional knowledge about the structure and content of the database, for example in the form of integrity constraints.

When an empty answer occurs just because of the absence of satisfactory data in the current database state, a solution consists in retrieving the data which are the closest wrt the initial query. In this paper, we will consider this second approach.

The remainder of this paper is organized as follows. In section 2, we briefly present the solutions proposed for query weakening in a Boolean framework, and we show the specific aspects of the weakening problem in a fuzzy querying framework. Then, in section 3, we describe a possible approach for fuzzy query weakening which is based on the use of linguistic modifiers and a semantic distance between queries. To conclude, we recall the main points of the paper and draw some working directions for the future.

2 Query Weakening

2.1 Boolean Query Weakening

Some weakening strategies for Boolean queries have been proposed in the context of cooperative answering. Considering that most user queries contain presuppositions, some of them false, Kaplan [11] proposed to correct the erroneous presuppositions instead of answering "none" (corrective answer). Janas improved and assessed this method using a predicate calculus formalism [10]. Cuppens and Demolombe [6] introduced the notion of loosening some of the constraints in a query to find answers close to those asked for (suggestive answers). Guyomard and Siroux [9] proposed a single mechanism to provide suggestive and corrective answers based on the use of a sub-question lattice.

A somewhat similar approach has been proposed by Motro [13] and implemented in the system FLEX. In this approach, if the answer is empty, the original query is passed to a query generalizer which issues a set of more general queries to determine whether the empty answer is genuine or whether it reflects erroneous presuppositions on behalf of the user (it then explains them).

Chu and Chen [5] have pursued this notion of close answers and proposed an approach called Neighborhood Query Answering, which is based on the concept of type abstraction hierarchy [4]. The objective is to provide generalized information relevant to the original query and within a certain semantic distance to the exact answer. Queries are rewritten by replacing relations and terms from the query with corresponding relations and terms from higher in the hierarchy.

Along similar lines, Gaasterland, Godfrey and Minker [8] have introduced a method called relaxation for expanding deductive database and logic programming queries. The relaxation method expands the scope of a query by relaxing the logical constraints implicit in the query.

Most of these approaches use a measure of "semantic distance" between queries (resp. between answers) in order to determine modified queries which are close to the original one (resp. to extend the answer with objects close to what is addressed in the original precise query).

In the Boolean framework, when the direct answer to a query is empty, rather than simply returning the null answer, the cooperating system may : i) exploit relevant domain knowledge as an explanation, guiding the user to formulate a user-modified query, ii) apply relevant domain knowledge to obtain a system-modified query. In-between these extreme principles where the system leaves all responsibility to the user (resp. does not involve the user at all), different schemes of interaction with the user arise, such as : iii) mastering a dialog with the user to obtain knowledge about user preferences and apply it to elaborate a system-modified query. Query weakening, i.e. system modification of an original query with a null answer, is what ii) and iii) are about. While ii) may be regarded as "pure" weakening, iii) is also weakening, but guided by what may be acquired in the user dialog (and potentially also what may be obtainable from a representation of knowledge about the user). Below, we restrict to ii), i.e. pure weakening. The general strategy for pure weakening of a query Q can be formulated in the following way : one must replace Q by the "best" modification Q' with properties : i) Q' is semantically close to Q, in the sense that elements in the answer of Q' still "fit" the user's intention, ii) Q' is a non null query (has a non empty answer). Two major problems are induced by these properties. First, since we cannot directly define a measure of "closeness" or "distance" from user intention, the main difficulty in query weakening is to capture this closeness by other means. The second problem is that of finding a non null modification of a null query. In general, it cannot be assumed that all the possible modified queries can be evaluated against the database.

2.2 About Fuzzy Query Weakening

Frameworks for fuzzy querying exploit cooperative behaviour in the sense that : i) domain knowledge (in the form of term definition) is reflected in answers, ii) frequency of empty answers is reduced when queries are not crisp. As a matter of fact, it shall be noted that the most simple and natural modification goes through the regulation of the threshold in queries explicitly requiring result calibration (queries restricted to a membership degree higher than 0.5 might be successfully weakened simply by reducing the threshold for instance to 0.3). In the following, we do not consider this simple case and we assume that an empty answer means that the 0-cut of the query is empty. In a fuzzy framework, any principle of weakening crisp queries may be applied, but fuzzy frameworks clear the ground for additional and in a sense more natural ways of weakening. Successful weakening of for instance "young and well-paid" might lead to "young and rather well-paid", the semantics of which should be obvious to a user acquainted with the meaning of "rather". Thus, in the weakening process, advantage may be taken of the intrinsic, but well-understood, "semantic distance" between a fuzzy term and its modification by a well-known modifier. Simple query weakening (as applied in the crisp case) by modification to sub-queries should of course also be well-understood by the user, but will in many cases turn too far away from the intention, e.g. if the user attaches equal importance to the two conditions in the query, then modification to the subquery "well-paid" cannot be satisfactory. Thus query weakening in the fuzzy framework is at the same time more exceptionally required and potentially more meaningful.

3 An Approach to Fuzzy Query Weakening

3.1 Introduction

As mentioned above, fuzzy queries reduce the risk of obtaining an empty answer. Nevertheless, they do not completely permit control of the number of elements in the answer. As a matter of fact, in the general case, a fuzzy query allows element discrimination only in certain portions of the domain. For instance, if we consider the predicate P represented below, only the elements belonging to the interval I1 or to the interval I2 can be discriminated by P (I1 and I2 are the portions where the degree is not constant).

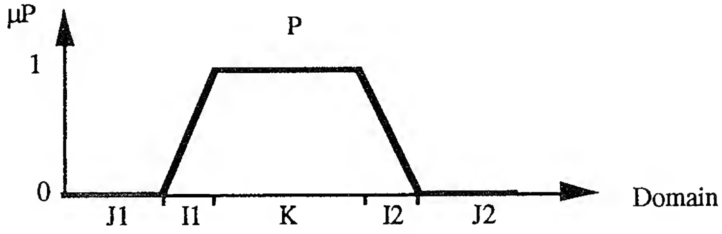


Fig. 1. Discrimination areas

In particular, when all the elements belongs to J1 $\cup$ J2 (complementary of the support of P), no discrimination can be performed, all the elements get a null degree and the answer is empty. When the query involves several fuzzy predicates linked by fuzzy connectors, the problem becomes more complicated. For example, the query "P(x) and Q(y)" can lead to an empty answer if all the pairs (x, y) are such that $\mu_P(x) = 0$ or $\mu_Q(y) = 0$ (the conjunction is interpreted as a minimum).

3.2. A simple case : single-predicate queries.

When the query only involves one fuzzy predicate, the only method to weaken the query is to apply a modifier on the fuzzy term. For example, the query "find the employees who are young" can be transformed into "find the employees who are more-or-less young". If we want more-or-less(P) to be a weakening of P, the linguistic modifier more-or-less must have two properties : i) it does not decrease the membership degree for any element of the domain, i.e. $\forall x \in$ Domain(A), $\mu_{\text{more-or-less(P)}}(x) \geq \mu_P(x)$ where A denotes the attribute concerned by P, ii) it extends the support of the fuzzy term, i.e. $\{x \mid \mu_P(x) > 0\} \subset \{x \mid \mu_{\text{more-or-less(P)}}(x) > 0\}$.

Most of the proposed modifiers do not have both properties. For example, modifiers defined as : $\mu_{\text{mod(P)}}(x) = \mu_{P^n}(x) = \mu_P(x)^n$ [16], or as : $\mu_{\text{mod(P)}}(x) = \mu_{P^n}(x) = (P\ \theta \ldots \theta\ P)(x)$ (θ, a non idempotent t-conorm for a dilatator, is applied n times) [15] do not extend the support of P. Modifiers defined in terms of translations [3] are not satisfactory either since they do not verify the first property. In [2], B. Bouchon-Meunier introduces a new family of modifiers, denoted m, applying to a fuzzy term f. This term is seen as a possibility distribution defined by five parameters (A, B, a, b, ε) such that $f(x) = \varepsilon$ if $x \leq A - a$ or $x \geq B + b$, $f(x) = 1$ if $A \leq x \leq B$, and $f(x) = f'(x)$ if $A - a \leq x \leq A$, $f(x) = f''(x)$ if $B \leq x \leq B + b$, for two functions f' and f'' respectively non-decreasing for $x \leq A$ and non-increasing for $x \geq B$, with $f'(A) = f''(B) = 1$ and $f'(A - a) = f''(B + b) = \varepsilon$. The term f is associated with a function φ such that $\varphi(x) = 1$ if $A \leq x \leq B$, $\varphi(x) = f'(x)$ if $x \leq A$ and $\varphi(x) = f''(x)$ if $x \geq B$. The family of modifiers m is introduced, yielding the following possibility distribution on the universe U : $\forall x \in U$ $g(x) = m(f(x)) = \min(1, \max(0, \alpha\varphi(x) + \beta))$ for two real valued parameters α and β, the attributes modified by m corresponding to parameters (A', B', a', b', ε'). Belonging to this family of modifiers, one called ν-rather is of particular interest for our purposes. It corresponds to values $\alpha = \nu \in [1/2, 1[$ and $\beta = 1 - \nu$, such that

$\forall x \in U \;\; g(x) = \max(0, \nu\varphi(x) + 1 - \nu)$ corresponding to parameters (A, B, a', b', ε) with a' > a, b' > b.

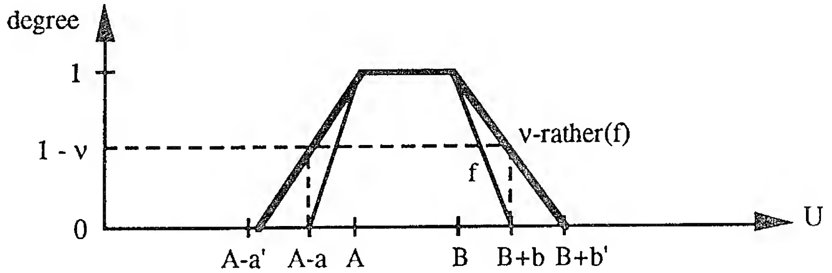


Fig. 2. Behaviour of the modifier ν-rather

This modifier preserves the specificity of the attribute, which means that f and g are equal to 1 for the same elements of U, but that f is null on a subset of U strictly containing the part of U where g itself is null. As this modifier obviously has the two properties required for query weakening and as its definition is general enough, we will use it in the following as the typical example of weakening modifier.

Thus, a simple strategy can be defined for the weakening of a query Q involving only one predicate P(x). In case of an empty answer, the query Q is transformed into Q_1 = rather(P)(x) and the process can be repeated n times until the answer to the question Q_n = rather (rather (... rather (P) ...))(x) is not empty. This strategy for single-predicate queries includes an implicit measure of nearness : Q_i is nearer to Q than Q_j is if i < j. This weakening mechanism is very easy to implement, but a difficulty arises concerning its semantic limits. The system must know the maximum number of weakening steps (i.e. the number of times that a modifier is applied) that is acceptable according to the user, otherwise the final modified query Q_n could be too far, semantically speaking, from the original one. To solve this problem, one idea consists in asking the user to specify, along with his query, a fuzzy set F_P of forbidden values in the related domain. In this case, an element x would get a satisfaction degree equal to min $(\mu_{Q_i}(x), 1 - \mu_{F_P}(x))$ with respect to the modified query Q_i resulting of i weakening steps. The process ends when the answer is non-empty or when the complementary of the support of Q_i is included in the core of F_P (there is no need to continue because the answer will always be empty).

3.3 Weakening of Complex Fuzzy Queries

Introduction. When a query involves several fuzzy predicates linked by connectors, different weakening methods can be envisaged. Apart from modifications applying fuzzy linguistic modifiers, such as "rather", the fuzzy framework also allows the application of connector replacement, for instance by not taking the conjunction in a query too strictly, but still not replacing it by a disjunction. We will not consider this approach here for the sake of brevity. As far as the weakening strategy itself is concerned, two options can be studied : i) global query modification (acting on the entire query), ii) local query modification (acting on subqueries). The former has the advantage of being simple but conflicts somewhat with our aim, that is, to find the closest modifications. Sometimes, the cause of the empty answer resides in a part of the query and then it is not necessary to modify all the predicates. In such cases, local query modification seems more suitable. This can consist in applying a weakening modifier to a subquery or a term. Since global modification can be seen as a special case of local modification, we will particularly focus on this latter option.

Term modification. In this study, we only consider conjunctive queries, for the sake of simplicity. Given a single modifier r (rather) and a conjunctive query Q = P_1 and P_2 and ... and P_k, the set of modifications of Q by r is : $\{r^{n_1}(P_1)$ and $r^{n_2}(P_2)$ and ... and $r^{n_k}(P_k)\}$ where $n_i \geq 0$ and r^{n_i} means that the modifier r is applied n_i times. A problem is defining a semantic distance between queries. Intrinsic partial ordering is defined as : Q' < Q" if Q" can be obtained by applying one or more weakening steps to Q'. This partial ordering can be expected to be well understood by the user. To extend it into a total ordering, a simple solution is to count the number of applications of the modifier r : Q' < Q" if count(r in Q') < count(r in Q"). For this ordering to be acceptable, it must be assumed that the user attaches equal importance to all conditions. In addition, the modifier r must have the property of "equal weakening effect" on all terms. A possible definition of this property is : for each predicate P_i of the initial query, the increase of the cardinality of the support relatively to the domain of P_i must be of the same magnitude when a certain modifier r is applied to P_i. To ensure that the weakening modifier has the "equal weakening effect" property is in fact not the responsibility of the system, but rather of the domain expert. For Bouchon-Meunier's ν-rather ensuring the property is a matter of adjusting term-dependent parameters for the modifier. Let us denote inc(P, r(P)) the relative increase of the cardinality of the support when r is applied to P. We have :

$$\mathrm{inc}(P, r(P)) = (|\mathrm{support}(r(P))| - |\mathrm{support}(P)|) / C$$

where C is the cardinality of the "authorized" portion of the domain i.e. of the complementary of the core of F_P. Given n predicates $P_1, ..., P_n$, the equal weakening effect property for a set of modifiers $(r_{v_1}, ..., r_{v_n})$ can be defined as :

$$\mathrm{inc}(P_1, r_{v_1}(P_1) = \mathrm{inc}(P_2, r_{v_2}(P_2)) = ... = \mathrm{inc}(P_n, r_{v_n}(P_n)).$$

The total ordering induced by the semantic distance based on such a modifier determines a lattice of modified queries. For example, the lattice (reduced in Fig. 3 to two levels) associated with the weakening of the query "P_1 and P_2" will be :

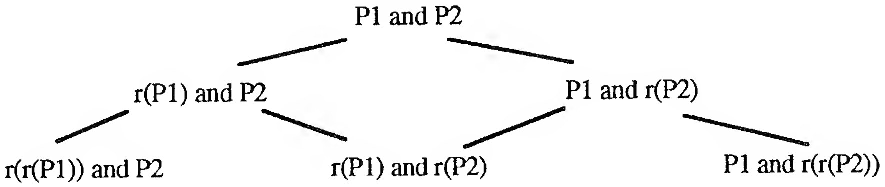


Fig. 3. Lattice of weakened queries

The system must then scan this lattice in a breadth-first manner in order to find the query with a non-empty answer which is as close as possible to the initial query.

To define a more efficient weakening strategy, we need additional knowledge to guide through the lattice arising from the ordering. We should like to be able to obtain an indication of the size of the answer - a measure of selectivity - of each weakened query, without evaluating it against the entire database. Selectivity defines a second ordering of modifications and the task in building a strategy is then to combine the semantic order with the selectivity order.

Selectivity is a concept usually employed in the database domain [14]. It expresses the proportion of tuples in a relation which satisfy a given condition. In a Boolean querying framework, we have sel(P) = card({t ∈ R | P(t) is true})/card(R) where R is the relation (or the Cartesian product of the relations) concerned by P. In a fuzzy framework, selectivity may be defined as : $\text{sel}(P) = \Sigma_{t\in R}\, \mu_P(t)/\text{card}(R)$. If the distribution of values in R is available, we can use :

$$\text{sel}(P) = \sum_{x \in \text{support}(P)} \text{card}(x, R.A) * \mu_P(x) \;/ \sum_{x \in \text{domain}(A)} \text{card}(x, R.A)$$

where card(x, R.A) means the number of tuples of R with the value x for the attribute A (argument of P). Since our main goal is to weaken conditions when empty answers occur, we can choose to focus on the membership to the support of a term rather than on the overall satisfaction of the term (we need only to know the proportion of tuples that obtain a non-zero degree for P). Hence, we can use a simplified expression :

$$\text{sbs}(P) = \sum_{x \in \text{support}(P)} \text{card}(x, R.A) \;/ \sum_{x \in \text{domain}(A)} \text{card}(x, R.A)$$

where sbs stands for support-based selectivity. When the distribution of A-values in R is uniform, we get : sbs(P) = card(support(P)/card(domain(A)) which can be computed only by considering the membership function of P.

In the framework of fuzzy query weakening, selectivity can be used to define a heuristic in order to determine the terms which must be modified first. More precisely, one can use selectivity for three purposes : i) detecting the terms P_i with an empty support ($\text{sbs}(P_i) = 0$), ii) detecting the conjunctions of terms applying to the same attribute with non intersecting supports, iii) choosing the predicate to be modified that provides the best global selectivity gain.

To illustrate case ii, let us take the case of a query with empty answer involving two terms restricting the same attribute, for instance, "$P_1(A)$ and $P_2(B)$ and $P_3(B)$". Let us suppose now that P_2 and P_3 are conflicting in the sense that the intersection of their support (i.e. the support of "P_2 and P_3") is empty. Then, all modifications obtained by modifying P_1 one or more times ($r^n(P_1(A))$ and $P_2(B)$ and $P_3(B)$, with $n \geq 1$) will also receive an empty answer. To cope with this problem, support conflicting terms must be given high priority when choosing what to modify.

As regards case iii, an idea consists in defining a measure of gain that will indicate the interest of each possible weakening in terms of increasing the global selectivity. In some but not in all cases, selectivity of a query can be closely estimated from selectivity of the predicates in the query. For the query $Q = P_1$ and P_2 ... and P_n, if all P_i's refers to distinct attributes, if the available attribute value distributions correspond to the actual distributions and if no pair P_i, P_j refers attributes with correlated values, then $\mathrm{sel}(Q) = \mathrm{sel}(P_1) * \ldots * \mathrm{sel}(P_n)$. Thus, we can define a measure of gain as :

$$\mathrm{gain}(Q, Q') = \mathrm{sbs}(Q') - \mathrm{sbs}(Q) = \prod_{i=1}^{n} \mathrm{sbs}(r^{k_i}(P_i)) - \prod_{i=1}^{n} \mathrm{sbs}(P_i)$$

where Q' is the query obtained from Q by weakening r^{k_i} times the predicate P_i.

Distributions may, but do not have to, be explicitly specified to the system. For an unspecified single-dimensional distribution, we take the uniform distribution over the domain of concern as default. For an unspecified multi-dimensional distribution of values for two or more attributes we assume no correlation and infer the multi-dimensional distribution from the distributions of the participating attributes. Adding explicitly specified distributions to the system simply corresponds to a refinement of selectivity computation.

If two or more predicates in a query refers to the same attribute, we may potentially, as mentioned, reveal a conflict - by an empty intersection of the support of these predicates. Therefore the predicates appearing in a query must be grouped according to the attributes referred and the gain of selectivity in modifying a predicate must be measured by the gain for the group in which the predicate participates. Thus, a query such as "$P_1(A)$ and $P_2(B)$ and $P_3(A)$", with two predicates referring to the predicate A, will have two groups, as indicated by square brackets in the rewritten version : "[$P_1(A)$ and $P_3(A)$] and [$P_2(B)$]". While the global gain in modifying P_2 is $(\mathrm{sbs}(r(P_2)) - \mathrm{sbs}(P_2)) * \mathrm{sbs}(P_1$ and $P_3)$, the gain in modifying for instance P_1 is $(\mathrm{sbs}(r(P_1)$ and $P_3) - \mathrm{sbs}(P_1$ and $P_3)) * \mathrm{sbs}(P_2)$. This will ensure a zero gain if the intersection of the support of $r(P_1)$ and P_3 remains empty. If explicitly specified multidimensional distributions are available, we must group predicates, not only in groups referring to the same attribute, but also in groups corresponding to these distributions. We shall restrict to a disjoint final set of groups simply by only letting a multidimensional distribution on a set of attributes justify a group if none of the referred attributes is already referred by another group.

The measure of gain gives us a basis to define a weakening strategy. Query modifications must be generated in a breadth-first manner if we want the resulting query to be as semantically close as possible to the initial one. Let us consider the ith level of the lattice of weakened queries (corresponding to the ith weakening step). The gain is evaluated for each weakened query of this level and the queries are then ordered in decreasing order according to their measure of gain. The "best" weakened query (with the highest value of gain) is processed. If its answer is empty, the second one is processed and so on. If all the answers are empty, the weakened queries belonging to the level i+1 are generated and the same method is applied. Our general principle below weakens a query Q already divided in groups as described above: $Q = G_1$ and ... and $G_k = P_1$ and ... and P_n . If no weakened query with a non-empty answer can be found, "false" is returned, otherwise the "closest" weakened query with a non-empty answer, when processed against the current database, is returned. Step 2) processes the groups in Q until all of these have positive selectivity. In this processing, weakening is performed individually on groups.

QUERY WEAKENING PRINCIPLE

```
1) $Q' = G_1$ and ...and $G_k = P_1$ and ... and $P_n$
2) while for some i in {1, ..., k} $sbs(G_i) = 0$
        set $G'_i$ = weaken($G_i$)
        if $G'_i$ = false then return false STOP
                  else replace $G_i$ by $G'_i$ in Q'
3) if $Q' \neq Q$ and actual-cardinality(Q') > 0 then return Q'
   else return weaken(Q')
```

We assume below that the lattice of weakened queries from a query Q incorporates the maximum number of allowed applications of the modifier r. Thus the lattice is finite and includes only allowed modifications of Q.

WEAKENING FUNCTION

```
weaken(Q)
1) $l = 1$
2) let MQ = {$Q'_1$, ..., $Q'_m$} be the queries at level $l + 1$ in the lattice from Q
3) if MQ is empty then  return false STOP    /* no weakened query exist

4) in order of decreasing gain(Q, $Q'_i$) do
        if gain(Q, $Q'_i$) > 0 then
                if actual-cardinality($Q'_i$) >0 then
                        return $Q'_i$  STOP    /* a weakened query is found
5) $l = l + 1$
6) go to 2)
```

The function "actual-cardinality(Q)" returns the number of tuples in the answer to Q, when processed against the current database. During the evaluation of each weakened query, the set of forbidden values for each attribute, if it exists, must also be taken into account.

4 Conclusion

In this paper, we have dealt with cooperative answering in the context of usual database fuzzy querying. More precisely, our approach is intended to avoid empty answers to fuzzy relational queriès. We have defined a weakening mechanism generating queries more tolerant than the initial user one, thus solving the problem at least partially. The proposed strategy consists in relaxing one or several fuzzy predicates involved in the query, by using a fuzzy linguistic modifier which extends the support of a term. To be acceptable, this modifier must have an equal weakening property i.e. it must increase the cardinality of the support of each term with the same magnitude. We propose using such a modifier to build a lattice of weakened queries expressing a semantic distance between queries. A second ordering, based on selectivity, is then used as a heuristic to improve the weakening strategy.

This work is a preliminary study in the sense that some experimentations should now be carried through to evaluate the efficiency of the proposed strategy. Several aspects remain open, in particular the interaction between the weakening system and the user. Another perspective concerns the capability of the system of providing informative answers to explain the cause of the empty answer, for example by using general fuzzy statements describing the current database state.

References

1. P. Bosc, O. Pivert: Some Approaches for Relational Databases Flexible Querying, J. of Intell. Inf. Syst., 1, 323-354 (1992)
2. B. Bouchon: Stability of linguistic modifiers compatible with a fuzzy logic, Uncertainty in Intelligent Systems, LNCS, 313, 63-70 (1988)
3. B. Bouchon-Meunier, J. Yao: Linguistic modifiers and imprecise categories, J. of Intell. Syst., 7, 25-36 (1992)
4. W.W. Chu, Q. Chen, R. Lee: Cooperative Query Answering via Type Abstraction Hierarchy, in S.M. Deen ed., Cooperating Knowledge Based Systems, New York : Elsevier, 271-292 (1990)
5. W.W. Chu, Q. Chen: Neighborhood and Associative Query Answering, J. of Intell. Inf. Syst., 1, 355-382 (1992)
6. F. Cuppens, R. Demolombe: Cooperative Answering : a methodology to provide intelligent access to Databases, in L. Kerschberg ed., Proc. of the Second International Conference on Expert Database Systems, 333-353 (1988)
7. T. Gaasterland, P. Godfrey, J. Minker: An Overview of Cooperative Answering, J. of Intell. Inf. Syst., 1, 2, 123-157 (1992)
8. T. Gaasterland, P. Godfrey, J. Minker: Relaxation as a Platform for Cooperative Answering, J. of Intell. Inf. Syst., 1, 3/4, 293-321 (1992)
9. M. Guyomard, J. Siroux: Suggestive and Corrective Answers : A Single Mechanism, in The Structure of Multimodal Dialogue, M.M. Taylor et al. eds, Elsevier Sc. Pub. (North Holland), 361-374 (1989)
10. J.M. Janas: On the Feasibility of Informative Answers, in Advances in Database Theory, H. Gallaire et al. eds., Plenum Press, 397-414 (1981)
11. S.J. Kaplan: Cooperative Responses from a Portable Natural Language Query System, Artificial Intelligence, 19, 2, 165-187 (1982)
12. A. Motro: SEAVE : A Mechanism for Verifying User Presuppositions in Query Systems, ACM Trans. on Off. Inf. Syst., 4, 4, 312-330 (1986)
13. A. Motro: FLEX : A Tolerant and Cooperative User Interface to Databases, IEEE Trans. on Knowledge and Data Engineering, vol. 2, n° 2, 231-246 (1990)
14. G. Piatetsky-Shapiro, C. Connell: Accurate estimation of the number of tuples satisfying a condition, ACM-SIGMOD, 256-276 (1984)
15. C. Yong-Yi: An approach to fuzzy operators, Busefal, 9, 59-65 (1981)
16. L.A. Zadeh: A fuzzy-set theoretic interpretation of linguistic hedges, J. of Cyb., 2, 4-34 (1972)

Transforming Queries from a Relational Schema to an Object Schema: A Prototype Based on F-logic*

Yahui Chang and Louiqa Raschid and Bonnie Dorr

Department of Computer Science, University of Maryland, College Park, MD 20742

Abstract. This paper describes a technique to support interoperable query processing when multiple heterogeneous knowledge servers are accessed. The problem is to support query transformation transparently, so a user can pose queries locally, without the need of global knowledge about different data models and schema. In this paper, we focus on transforming SQL source queries, posed against a relational schema, to XSQL queries to be evaluated against an object schema, so that information can be shared. We will describe an extraction algorithm which extracts semantics from a source SQL query, and the query transformation algorithm, described by example mapping rules, which access mapping knowledge among the schema/models and the query languages. This knowledge is represented in a parameterized canonical form, using a second-order logic, namely F-logic.

1 Introduction

One of the most important requirements of a heterogeneous information system is to support interoperable query processing when multiple heterogeneous servers are being accessed. The problem is to support query transformation transparently, so a user can pose queries locally, without needing global knowledge about different data models and schema. After such a transformation, a query may be answered using information from multiple sources. We present a technique to achieve interoperability, through capturing knowledge about source and target query languages, data models, schema and mapping information, in local and mapping knowledge dictionaries, and applying a set of mapping rules to obtain a transformation. We represent the dictionary knowledge and mapping rules using a second-order logic language, F-logic (see Kifer and Lausen (1989)). F-logic has good modeling constructs for representing mapping knowledge and it can represent object schema.

This research can be placed in the context of other research in the area of *representational heterogeneity*, where sharable knowledge is stored in multiple schema using different models, with possible mismatch between schemas and query languages. Many approaches advocate the use of a global schema (Ahmed

* This research has been partially supported by the Defense Advanced Research Project Agency under grant DARPA/ONR grant 92-J1929.

et al. (1993), Arens *et al.* (1992), Kim *et al.* (1993)). A disadvantage of the global schema approach is that a single global schema must be accepted by all users and queries are expressed against this schema in a global query language. This is not feasible in existing legacy environments, where an application already posed queries against a local schema. These local legacy applications may wish to access information from other servers, without posing new queries against the global schema. Our approach does not require schema integration among the multiple servers, and addresses the problem of transforming local queries transparently, so the local legacy application does not need explicit knowledge (about the global schema or mapping knowledge among the schema) to pose a query. The research that is closest to our work is described by Lefebvre *et al.* (1992) and Qian (1993), but it is limited to relational schema.

Our approach encapsulates the information on resolving representational heterogeneity in a knowledge dictionary and uses this to support query transformation. Users need not write their queries against a global schema since we provide interoperability in a transparent manner. In this paper, we address the issue of transforming SQL queries posed against a relational schema to produce XSQL queries[2] against the object schema. This paper is organized as follows: section 2 has examples of transforming SQL queries to XSQL queries. Section 3 provides an architecture for interoperable query processing. Section 4 describes the extraction algorithm which represents a source SQL query in a canonical form. Section 5 describes the F-logic implementation which transforms the canonical SQL query to an XSQL query. The extraction algorithm has been implemented in Lex, Yacc and C. The mapping rules use an F-logic interpreter (Lawley, 1993).

2 Examples of Transforming Queries from Relational to Object Schema

In the relational schema (Figure 1), relation Register has a primary key, Ssn, and represents students who are currently registered. GradStudent and UGStudent also have Ssn as the primary key. Four other relations correspond to graduate and undergraduate students in two departments; they use Name as the primary key and attribute WorksIn refers to a project in which a student participates. The relation Project has a key Pno, and PI is a principal investigator.

In the object schema,[3] each node is an object. The dotted line represents a class hierarchy, and a solid arc represents a pointer to another object. In object schema #1 (Figure 2), Student-class has two immediate sub-classes Grad-class and UG-class, each of which has two sub-classes, identifying the Cs and Math students (4 classes). In object schema #2 (Figure 3), CsStudent-class and MathStudent-class are sub-classes of Student-class, and each of these classes has two sub-classes, for graduate and undergraduate students, respectively.

[2] The detailed syntax of XSQL queries is described in Kifer *et al.* (1992).

[3] Each object in the schema is a *class* object which may be distinguished from the *individual* objects or instances.

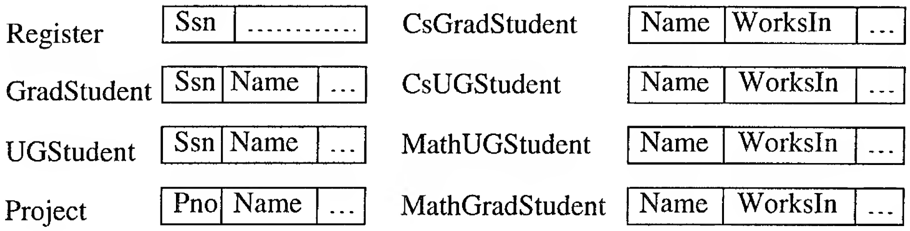


Fig. 1. Example relational schema

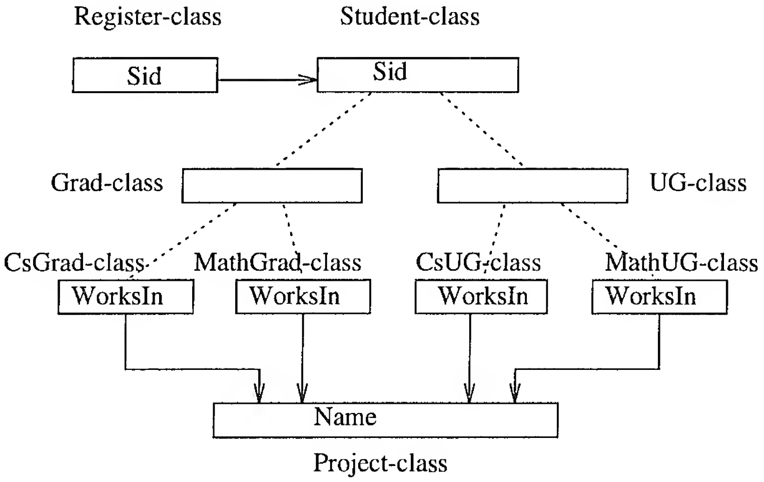


Fig. 2. Example object oriented schema #1

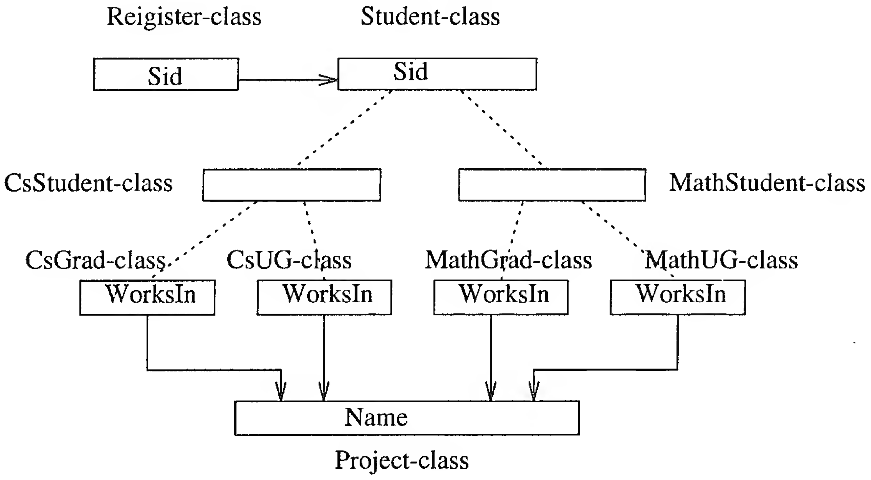


Fig. 3. Example object oriented schema #2

Queries against the relational schema will be expressed using SQL. To express a query against the object schema, we use an XSQL-like query (see Kifer *et al.* (1992)). We use simple XSQL queries that only differ from SQL queries in the path expression of the where clause. The *path expression* is as follows:

$$\mathrm{sel}_0.\mathrm{Att}_1\{[\mathrm{sel}_1]\}.\cdots.\mathrm{Att}_m\{[\mathrm{sel}_m]\}$$

A selector sel_i is an object identifier or an object selector and Att_i is an attribute. The attribute Att_{i+1} following each selector sel_i is either directly defined in the class of sel_i or it is an inherited attribute of sel_i and must be defined for one of the super-classes of sel_i. Each $sel_i.Att_{i+1}$ can either return a value or a pointer to an object. The selector sel_{i+1} indicates that it could be in some class specified by $sel_i.Att_{i+1}$, or it could be in any sub-class of this class.

Consider an SQL query to identify all principal investigators of the projects in which currently registered computer science graduate students conduct research, as follows:

Query 1 in relational schema

```
Select PI
from   Register, GradStudent, CsGradStudent, Project
where  Register.Ssn = GradStudent.Ssn and
       GradStudent.Name = CsGradStudent.Name and
       CsGradStudent.WorksIn = Project.Pno
```

In the object schema, these joins must be interpreted as selecting those students in CsGrad-class, and the projects in which they work. CsGrad-class occurs as an immediate sub-class of different classes in object schema #1 and #2. However, the equivalent XSQL query for these object schemas will be the same, since CsGrad-Class is a sub-class of Student-class in two schema and an XSQL query need not specify the different paths from Student-class to CsGrad-class. The query is as follows:

Query 2 in object schema #1 and #2:

```
Select Z.PI
from   Register-class X, CsGrad-class Y, Project-class Z
where  X.Sid[Y].WorksIn[Z]
```

3 Architecture for Query Transformation

Figure 4 illustrates the overall architecture of an Interoperability Module (IM) to support interoperable query processing. Corresponding to each heterogeneous DBMS there is a local knowledge dictionary that encodes (1) the query language constructs, and (2) the local schema, and it is accessed by the **Extraction Module (EM)** of the IM. The EM will represent the information extracted from the source query constructs in a canonical representation, e.g., CRrel for SQL. The **HeTerogeneous Mapping (HTM)** has several functions. The first function is identifying the mapping knowledge, HTMapping, in the mapping knowledge dictionary, representing the correspondence among entities in the different schema. For a mapping from a relational to an object schema, this knowledge is parameterized as HTrel-objMapping. The second function is identifying and applying the correct mapping rule, which determines the correct transformation. After applying the rules, HTM produces a TransfQuery, TQobj, to represent the target query in the canonical representation. The mapping rules are classified into different groups. When mapping from a relational to an object schema, the rules are as follows: **Ident-Class-Hier** (identifying class hierarchies), **Object-Ptr** (identifying pointers to objects), and **Exp-Join** (producing an explicit join pair). The final module of the IM is the **Generator Module (GM)** , and it uses knowledge on the cost of processing a query, to produce an optimal query in the target query language.

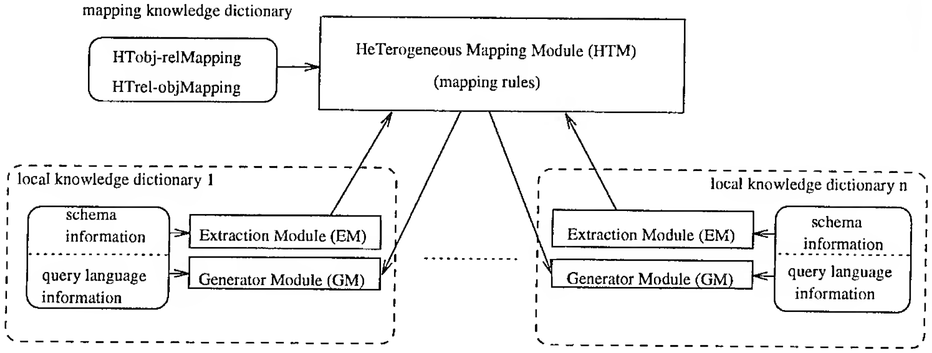

mapping knowledge dictionary
HTobj-relMapping
HTrel-objMapping
HeTerogeneous Mapping Module (HTM)
(mapping rules)
local knowledge dictionary 1
schema information
query language information
Extraction Module (EM)
Generator Module (GM)
local knowledge dictionary n
Extraction Module (EM)
Generator Module (GM)
schema information
query language information

Fig. 4. Architecture of the Interoperability Module (IM)

4 Extraction Algorithm

The extraction algorithm operates on a source relational query and produces a corresponding canonical structure, CRrel. There are several reasons for adopting such a structure instead of direct translation of a source query to the target query language form. First, different query languages (object or relational) have different constructs, e.g., path expression in XSQL, functional composition in OSQL, etc. In addition, we have the heterogeneity of the different schema themselves. Using a canonical structure allows us to separate the mapping knowledge due to the heterogeneity of schema from the knowledge needed to translate between the constructs in the different languages. If we use a direct translation approach, the rules will be very complicated because they have to resolve the schema conflicts and also consider the syntax of each language. This is precisely analogous to arguments for the use of an interlingual representation over a direct-mapping approach in the field of natural language translation (see, e.g., Dorr (1993)).

We use a data structure, CRrel, to represent the SQL query. For simplicity, we only discuss the constructs corresponding to the WHERE clause. The WHERE clause is used to express constraints and could involve complicated operations. We assume the WHERE clauses only consist of conjunctions, of the form: cond_1

AND cond_2 AND $\cdots$ cond_n. Each conjunct cond_i could be viewed as a predicate that is represented by comparing the attribute of relations with some other values, set-operation involving nested queries, etc.

The CRrel structure is represented as an F-logic data structure. Details of F-logic syntax are given in the next section.

CRrel[sub-rel ↔ (R_{sup}, R_{sub});
reference ↔ REF;
comparison ↔ (REL-OPR, OP1, OP2);
set-operation ↔ (SET-OPR, OP1, OP2)]

The values of the attribute *sub-rel* is a set of pairs with the form (R_{sup},R_{sub}), where the key of the relation R_{sub} is a subset of the key of the relation R_{sup}. The value of the attribute *reference*, REF, is a set of ordered lists of quadruples. Each quadruple has the form (R1,A1,R2,A2). This corresponds to the joining pair R1.A1 = R2.A2 in SQL, when there is a foreign key relationship between R1.A1 and R2.A2. The value of the attribute *comparison* is a set of triples with the form (REL-OPR, OP1, OP2), where REL-OPR is a relational operator, and OP1 and OP2 are operands. This corresponds to a comparison of the values of OP1 and OP2, and does not have special structural meanings. The last attribute *set-operation* corresponds to the operation involving sets. SET-OPR could be IN, or NOT IN, etc., which are used to test if the value of OP1 is IN or NOT IN the set represented by OP2. OP2 is a nested query and will be represented by another CRrel structure.

We now describe the portion of the algorithm for the EM which operates on a set of joining relation-attribute pairs. The aim is to consider each joining pair in the WHERE clause, and use schema information such as keys, key inclusion dependency, foreign keys, etc., to simplify the query, eliminate redundant pairs, and obtain relevant structure information. We use the key inclusion dependency to identify the minimal subsets of key attributes in a query, since this may correspond to the most specific subclass in an object schema. We build an *inclusive class* named by each minimal subset. We use the foreign key relationship to construct an ordered list which corresponds to referencing a sequence of objects.

Extraction Algorithm

- Input : a set of joining relation-attribute pairs
- Output : CRrel in F-logic form
  1. Build a graph where each node corresponds to a relation, and each edge is either an *inc-link* or a *ref-link* when applicable. The *inc-link* represents a key inclusion dependency, and the *ref-link* represents a foreign key relationship.
     For each input pair (R1.A1 = R2.A2), create new nodes for relations R1 or R2 if they have not been created previously. Then check if the inclusion dependency exists and create corresponding *inc-links* and *ref-links*:
     (a) If A1 and A2 are both keys, check if there exists a key inclusion dependency between them. If so, create an inc-link pointing from the node corresponding to the subset relation to the superset relation.

(b) If (R1.A1) is a foreign key of R2 which has the key A2, create a ref-link pointing from the node R1 to the node R2. Annotate the link with the quadruple (R1, A1, R2, A2) to denote the joining relation-attribute pairs. Also test if R2.A2 is a foreign key of relation A1.

(c) If the pairs do not have specific structural meanings, output the F-logic form: CRrel[comparison $\leftrightarrow$ ("=",R1.A1,R2.A2)]

2. Check inc-links and build *inclusive classes.*

Let each node which does not have any incoming inc-links in the graph be the root, say R_1. It represents the most specific subset in that hierarchy with respect to inclusion dependency. Traverse through the inc-links and include each visited node, say R_j, in the inclusive class denoted by R_i. Note that R_j may be in several inclusive classes if it has more than one incoming inc-links. The corresponding output structure is:

$$\text{CRrel[sub-rel} \leftrightarrow (\text{R}_j,\ \text{R}_i)].$$

If there is a cycle through the inc-links, all those corresponding relations $(R_1, \cdots R_n)$ will have the same set of values for their keys. It means they actually correspond to a vertically partitioned relation where the attributes of a tuple are distributed in several relations. We randomly pick one relation R_i as the name of the inclusive class and create the following constructs:

CRrel[sub-rel $\leftrightarrow$ (R_1, R_i)], $\cdots$, CRrel[sub-rel $\leftrightarrow$ (R_i, R_i)], $\cdots$,
CRrel[sub-rel $\leftrightarrow$ (R_n, R_i)].

For those nodes R_i which do not have incoming and outgoing inc-links, we will produce the following structure: CRrel[sub-rel $\leftrightarrow$ (R_i, R_i)].

3. Check the ref-links and produce ordered lists.

(a) This step is to modify the graph by removing the inc-links and "collapse" the nodes which are in the same inclusive classes.

For each node, say R, whose inclusive class, say C, is not equal to R:

for each outgoing ref-link n pointing from R to some R', add a new ref-link from C to the inclusive class C' of R', and delete n;

for each incoming ref-link n pointing to R from some R', add a new ref-link pointing to C from the inclusive class of R', and delete n.

(b) This step is to navigate through the ref-links and produce an ordered list which corresponds to a sequence of referential relationships.

Let each node which does not have incoming ref-links be a root. Traverse through the links from the root to each leaf to create an ordered list of joining pairs. Suppose there is a link pointing to the node N, with annotation (R1,A1,R2,A2), and a link pointing from the node N, with annotation (R3,A3,R4,A4), then output:

CRrel[reference $\leftrightarrow$ cons(cons(L,(R1,A1,R2,A2)),(R3,A3,R4,A4))],

where L (initially nil) is the output from the previous traversal.

5 Representation for Mappings

This section discusses the mapping rules, and HTrel-objMapping in the mapping knowledge dictionary. The rules are represented as an F-logic program.

5.1 F-logic

We borrow the brief description from Lefebvre *et al.* (1992) to introduce the syntax of F-logic. In F-logic, the *instance term* $o : c$ means that the object o is an instance of class c. A *data term* $o[m@a_1, \cdots, a_n \rightarrow v; m'@a_1, \cdots, a_p \leftrightarrow \{v', v''\}]$ means that the value of the functional method m' with arguments a_1 to a_p for the object o is a set containing the values v' and v''. If a method m has no arguments, "@" will be omitted. The symbol $\rightarrow$ indicates a single-valued method, and the symbol $\leftrightarrow$ indicates a set-valued method.[4] An object can be denoted by a constant, or a term. For example, *dept*(*cs*) is a valid object identifier. Notational conventions allow us to write $o[m' \leftrightarrow v]$ instead of $o[m' \leftrightarrow \{v\}]$ for a single element of a set-valued method; the expression $o[m \rightarrow v; n \rightarrow v']$ is equivalent to $o[m \rightarrow v] \wedge o[n \rightarrow v']$.

An F-logic program consists of a set of data declarations(data or instance terms), and a set of *deduction rules*. A deduction rule has a *head*, which is a data term, and a *body*, which is a conjunction of data and instance terms. Disjunction and negation are allowed in the body of rules.

5.2 Data Structures

There are three kinds of data structures used by the mapping rules described in section 4.3. CRrel is produced by the Extractor Module from an SQL query and has been described in a previous section. HTrel-objMapping captures the mapping information from a relational schema to an object schema D. The class hierarchy in D is represented by the method *super-class*. Applying the mapping rules and HTrel-objMapping, we produce TQobj which represents the transformed query in the object schema. Its method *sub-class* will return the corresponding class label for the inclusive class of R. The method *path* will produce an XSQL path expression, given an ordered list of joining pairs. The method *join* is invoked to produce an explicit join in XSQL format.

HTrel-objMapping(R, D)[mapped-class ↔ C[super-class ↔ C'];
    mappings ↔ map(R,A,D)[object-attr-domain ↔ (O,AN,OO)]]

TQobj(CRrel, D)[sub-class@R → O; path@J → L; join@J → K]

[4] This symbol is different from the one used in the original paper.

5.3 Mapping Rules

The mapping rules can be classified into three groups. The first group **Ident-Class-Hier** deals with *inclusive classes*. The second group **Object-Ptr** determines the appropriate path in the object schema corresponding to a given ordered list of joining pairs in the relational schema. The last group **Exp-Join** is used when an explicit join is needed. We now show some examples of the application of the rules. The object schema #2 in Figure 2 will be called d_2.

- **Group One: Ident-Class-Hier**
  The rules in this group determines the class name in the object schema, corresponding to the *inclusive class* associated with a relation. We must check if the class hierarchy in the target object schema corresponds exactly to the *inclusive classes* in the relational schema. If so, we produce an optimal query for the relevant (most specific) class. If there is no exact correspondence among the schema, we require an explicit join.
  Suppose the relation GradStudent is in the *inclusive class* denoted by CsGradStudent and the corresponding CRrel is: CRrel[sub-rel ↔ (GradStudent, CsGradStudent)]
  Also, CsGradStudent maps to the class CsGrad-class, and CsGrad-class is a subclass of Grad-class. Thus, the *inclusive class* hierarchy of the relation schema corresponds exactly to the subclass hierarchy in the object schema d_2. The instantiation of the mapping rules will be as follows:

  TQobj(CRrel,d_2)[sub-class@GradStudent → CsGrad-class] ←
      CRrel[sub-rel ↔ (GradStudent, CsGradStudent)] ∧
      HTrel-objMapping(GradStudent,d_2)[mapped-class ↔ Grad-class] ∧
      HTrel-objMapping(CsGradStudent,d_2)[mapped-class ↔ CsGrad-class
                                                  [super-class ↔ Grad-Class]]

- **Group Two: Object-Ptr**
  The rules in this group produce an XSQL-like path expression. The parameter of the method *path* of TQobj is an ordered list of quadruples. A quadruple (R1, A1, R2, A2) corresponds to a joining pair R1.A1 = R2.A2, where the ref-link is pointing from R1 to R2. If R1.A1 maps to O1.AN1, R2.A2 maps to O2.AN2, and the *inclusive class* of R1 maps to the subclass O3 of O1, and the *inclusive class* of R2 maps to the subclass O4 of O2, then the result of the method *path* will be the quadruple (O3,AN1,O2,O4). AN1 is an attribute of O3, which could be explicit or inherited attribute. O2 is the object referenced by the attribute AN1. O4 is a specific subclass of O2 and the XSQL query returns this subclass. The corresponding XSQL query is O3.AN1[O4]. Note that this rule uses the previous rule in group Ident-Class-Hier to get the value for the method sub-class of TQobj.
  Consider the joining pair Register.Ssn = GradStudent.Ssn. Observe that the relations Register and the relation GradStudent correspond to the classes Register-class and Grad-class respectively, so the instantiation of the mapping rules will be as follows:

TQobj(CRrel,d_2)[path@(Register,Ssn,GradStudent,Ssn) $\rightarrow$
(Register-class,Sid,Student-class,CsGrad-class)] $\leftarrow$
CRrel[reference $\leftrightarrow$ (Register,Ssn,GradStudent,Ssn)] $\wedge$
HTrel-objMapping(Register,d_2)[mappings $\leftrightarrow$ map(Register,Ssn,d_2)
[object-attr-domain $\leftrightarrow$ (Register-class,Sid,Student-class)]]$\wedge$
HTrel-objMapping(GradStudent,d_2)[
mappings $\leftrightarrow$ map(GradStudent,Ssn,d_2)
[object-attr-domain $\leftrightarrow$ (Grad-class,Sid,integer)]] $\wedge$
HTrel-objMapping(GradStudent,D) [
mapped-class $\leftrightarrow$ Grad-class[super-class $\leftrightarrow$ Student-class]] $\wedge$
TQobj(CRrel,d_2)[sub-class@Register $\rightarrow$ Register-class] $\wedge$
TQobj(CRrel,d_2)[sub-class@GradStudent $\rightarrow$ CsGrad-class]

References

Ahmed, R., J. Albert, W. Du, W. Kent: An overview of Pegasus. Proceedings of the International Conference on Data Engineering. (1993)

Albert, J., R. Ahmed, M. Ketabchi, W. Kent, M. Shan: Automatic importation of relational schemas in Pegasus. Proceedings of the International Conference on Data Engineering. (1993)

Arens, Y. and C. Knoblock: Planning and reformulating queries for semantically-modeled multidatabase systems. Proceedings of the International Conference on Knowledge Management. (1992)

Dorr, Bonnie J.: Machine Translation: A View from the Lexicon. MIT Press. Cambridge, MA. (1993)

Kifer, M. and G. Lausen: F-logic: A higher-order language for reasoning about objects, inheritance and scheme. Proceedings of the ACM Sigmod Conference. (1989)

Kifer, M., W. Kim, and Y. Sagiv: Querying object-oriented databases. Proc. of the ACM Sigmod Conference. (1992)

Kim, W., Choi, I., Gala, S. and Scheevel, M.: On resolving heterogeneity in multidatabase systems. Distributed and Parallel Databases, Vol. 1. (1993) 251–279

Lawley, M.: A Prolog interpreter for F-logic. Technical report. Griffith University. (1993)

Lefebvre, A., P. Bernus and R. Topor: Query transformation for accessing heterogeneous databases. Joint International Conference and Symposium on Logic Programming, Workshop on Deductive Databases. (1992)

Qian, X.: Semantic interoperation via intelligent mediation. Workshop on Research Issues in Data Engineering. (1993)

Raschid, L., Chang, Y. and B. Dorr: Interoperable Query Processing with Multiple Heterogeneous Knowledge Servers. Proceedings of the Second International Conference on Information and Knowledge Management. (1993)

Raschid, L., Chang, Y. and B. Dorr: Query Transformation Techniques for Interoperable Query Processing in Cooperative Information Systems. Proceedings of the Second International Conference on Cooperative Information Systems. (1994)

Knowledge Discovery in Databases: A Rule-Based Attribute-Oriented Approach

David Wai-lok Cheung[1]*, Ada Wai-Chee Fu[2] and Jiawei Han[3]**

[1] Department of Computer Science, The University of Hong Kong, Hong Kong.
[2] Department of Computer Science, Chinese University of Hong Kong, Hong Kong.
[3] School of Computing Science, Simon Fraser University, Canada.

Abstract. An attribute-oriented induction has been developed in the previous study of knowledge discovery in databases. A concept tree ascension technique is applied in concept generalization. In this paper, we extend the background knowledge representation from an unconditional non-rule-based concept hierarchy to a rule-based concept hierarchy, which enhances greatly its representation power. An efficient rule-based attribute-oriented induction algorithm is developed to facilitate learning with a rule-based concept graph. An information loss problem which is special to rule-based induction is described together with a solution suggested.

Keywords: learning and adaptive systems, knowledge discovery in databases, rule-based concept generalization, attribute-oriented induction.

1 Introduction

Knowledge Discovery is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [2]. Over the past twenty years, huge amounts of data have been collected and managed in relational databases by industrial, commercial or public organizations. This has created a need and a challenge for extracting knowledge from these databases. Knowledge extracted can be used to support deductive system or intelligent query answering system [6, 7]. Researches have been done with different approaches to tackle this problem [2].

In the previous studies [5], a **Basic Attribute-Oriented Induction** (**Basic AO Induction**) method has been developed for knowledge discovery in databases. It has been implemented in the database learning system **DBLearn** [5]. Both the applicability and performance of the system are encouraging.

A key in Basic AO Induction is the *attribute-oriented concept tree ascension* for concept generalization, a concept hierarchy is used to represent background

* Email: dcheung@csd.hku.hk

** The research of the third author was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada and the Centre for Systems Science of Simon Fraser University.

knowledge supplied by domain experts. The most commonly used concept hierarchies are *concept tree* and *lattice*, in which concepts are generalized to higher level concepts *unconditionally*. For example, in a concept tree defined for the attribute *GPA* of a student database, a 3.6 GPA (in a 4 points system) can be generalized to a higher level concept, perhaps to the concept *excellent*. This depends only on the GPA value but not any other information of a student. However, in some cases, domain experts (users) may need to represent the background knowledge in such a way that concept generalization would depend not only on the concept itself but also on other conditions. For example, in some institutions, they may want to apply different rules to different types of students. The same 3.6 GPA may only deserve a *good, if the student is a graduate*; and it may be *excellent, if the student is an undergraduate*.

This suggests that a more general concept generalization scheme should be *conditional* or *rule-based*. To this end, we propose a rule-based hierarchy for background knowledge representation called **Rule-Based Concept Graph**.

The adoption of rule-based concept graph substantially increase the representation and induction power of attribute-oriented generalization. To support this model, a **Rule-Based Attribute-Oriented Induction (Rule-Based AO Induction)** has been designed. Both the general principles and an efficient method for Rule-Based AO Induction are described in this paper.

The paper is organized as follows. The primitives of knowledge discovery and the principle of Basic AO Induction are briefly reviewed in Section 2. The general notions of Rule-Based Concept Generalization and Rule-Based Concept Hierarchy are discussed in Section 3. The principles of Rule-based AO induction is presented in Section 4. Applications of rule-based knowledge discovery are discussed in Section 5.

2 Basic Attribute-Oriented Induction

Task-relevant data, background knowledge, and *expected representation of learning results* are the three primitives that specify a learning task in Basic AO Induction [5]. A query that specifies a learning task is used to retrieve the task-relevant set of data from a database. In Basic AO Induction, the retrieved task-relevant tuples are stored in an **initial relation**, and concept tree is being used to represent background knowledge. Many kinds of rules, such as *characteristic rules*, and *discriminant rules* can be discovered by induction processes [5]. A tuple in a relation can be viewed as a logic formula in conjunctive normal form, and a relation is characterized by a large set of disjunctions of such conjunctive forms [3, 7]. Thus, both the data for learning and the rules discovered can be represented in either relational form (tuples) or first-order predicate calculus.

The fundamental idea in AO Induction is to generalize the initial relation to a **prime relation** and then to the **final relation**. During this process, attribute values in the relation are converted iteratively to higher level attribute values with respect to generalization paths in a concept tree. Tuples that have the same higher level attribute values are merged to form higher level tuples. The prime relation is generated when the number of distinct values of each attribute in the relation is less than the predefined **desirable attribute threshold**. Then the prime relation will be reduced further by selectively generalize some attributes until the size of the final relation is less than the predefined **generalization relation threshold** [4, 5]. At the end, tuples in the final relation which represent certain regularities are converted into knowledge rules. Note that the two notions *attribute value* and *concept* are used interchangeably here. When a generalized tuple is converted into a rule, attribute values (generalized) are converted into attribute-value pairs (predicates) in the rule which are intrepreted as high level concepts.

3 Rule-Based Concept Generalization and Concept Hierarchies

Concept generalization is a key component in database inductive learning. Concepts are ordered in a concept hierarchy by levels from specific (lower level) to general (higher level). Generalization is achieved by ascending concepts along the paths of a concept hierarchy.

In general, a concept can ascend via more than one path. A **generalization rule** can be assigned to a path in a concept hierarchy to determine whether a concept can be generalized along that path. For example, the generalization of a student's GPA in (2.0 – 2.49) can be performed along two paths with the following two generalization rules:

If a student's GPA is in the range (2.0 – 2.49) and he is an undergrad, then it is an average GPA.

If a student's GPA is in the range (2.0 – 2.49) and he is a grad, then it is a poor GPA.

Concept hierarchies whose paths have associated generalization rules are called **rule-based concept hierarchies**, and there are three types of rule-based concept hierarchies. In the first type, **deductive generalization rules** are associated with the generalization paths. A deductive generalization rule has the form: $A(x) \wedge B(x) \longrightarrow C(x)$. For a tuple x, concept (attribute value) A can be generalized to concept C (higher level attribute values) if condition B can be satisfied by x. The condition $B(x)$ can be a simple predicate or a very complex logic formula involving different attributes and relations. This type of hierarchy is called **deductive-rule-based concept graph**. In the second type, **computational generalization rules** are associated with the hierarchy. Each rule is represented by a condition which is value-based and can be computed from an attribute or a tuple or the database. The truth value of the condition would then determine whether a concept can be generalized via a path. The third type is a **hybrid hierarchy** in which either one of the above two types of rules can be associated with a path. In the scope of database induction, the same technique can be used on all the different types of rule-based hierarchies. Therefore, in the rest of this paper, we will use deductive-rule-based concept graph as a typical concept hierarchy.

4 Rule-Based Attribute-Oriented Induction

A rule-based AO induction system is defined by (EDB, RCH, DS, TS, KRS, ATh, RTh). EDB is the underlying extended relational databases, RCH is a set of deductive-rule-based concept hierarchies, one for each attribute, DS is a deductive system to support concept generalization, it includes all the deductive rules in RCH and a set of supporting rules. TS is a set of task specifications and KRS is a scheme, which is the first order predicate calculus, to represent knowledge. ATh and RTh are respectively the *desirable attribute threshold* and the *generalization relation threshold*. The goal of the induction system (ERD,RCH,DS,TS,KRS,ATh,RTh) is to extract general and useful knowledge from data in ERD that satisfy a task specification in TS by doing generalization along the direction provided by RCH with the support of DS. The learned result will be represented by the scheme defined by KRS.

Example 1. In this example, the EDB is a university student database defined by the schema *Student(Name, Status, Sex, Major, Age, Birthplace, GPA)*. The learning task in Ts is to discover the characteristic rules for CS students. The concept graph in RCH for the attribute GPA is defined in Fig. 1 and the associated deductive rules defined in Fig. 2. An equation $A \wedge B \rightarrow C$ in Fig. 2 represents the conditional rule: *If x satisfies condition B, then its attribute value A can be generalized to the higher level concept C.* For example, both R_6 and R_7 denote the rules:

R_6: *If a student's GPA is in (3.49 – 3.8) and (s)he is a graduate, then it can be classified as a good GPA.*

R_7: *If a student's GPA is in (3.49 – 3.8) and (s)he is an undergrad, then it can be classified as an excellent GPA.*

During the generalization process, one of these two rules would be applied to generalize the GPA in the range (3.49 – 3.8) to either *good* or *excellent*. □

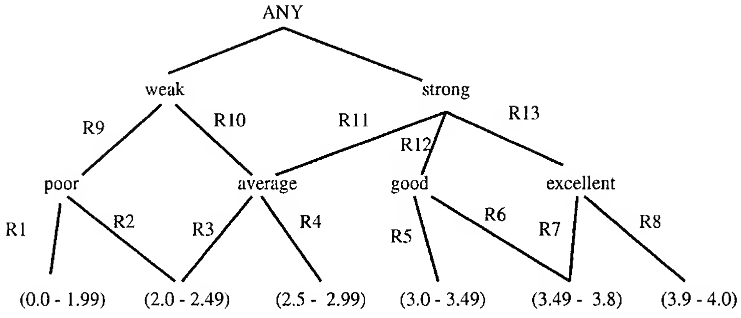


Fig. 1. A rule-based concept graph for *GPA*

$R_1 : \{0.0 - 1.99\} \rightarrow poor$
$R_2 : \{2.0 - 2.49\} \wedge \{graduate\} \rightarrow poor$
$R_3 : \{2.0 - 2.49\} \wedge \{undergraduate\} \rightarrow average$
$R_4 : \{2.5 - 2.99\} \rightarrow average$
$R_5 : \{3.0 - 3.49\} \rightarrow good$
$R_6 : \{3.49 - 3.8\} \wedge \{graduate\} \rightarrow good$
$R_7 : \{3.49 - 3.8\} \wedge \{undergraduate\} \rightarrow excellent$
$R_8 : \{3.9 - 4.0\} \rightarrow excellent$
$R_9 : \{poor\} \rightarrow weak$
$R_{10} : \{average\} \wedge \{senior, graduate\} \rightarrow weak$
$R_{11} : \{average\} \wedge \{freshman, sophomore, junior\} \rightarrow strong$
$R_{12} : \{good\} \rightarrow strong$
$R_{13} : \{excellent\} \rightarrow strong.$

Fig. 2. Conditional Generalization Rules for GPA

4.1 Induction Phase One : From Initial Relation to Prime Relation

The goal of the first phase is to generalize and reduce the initial relation to the prime relation. We describe here an efficient two passes algorithm for inducing the prime relation from the initial relation.

First Pass: Determination of the Desirable Level. The goal of this pass is to make use of the deductive system to find out, for each attribute A, the **minimum desirable level** [4] L_A in an efficient way.

In one scan of the initial relation, for each attribute A, the deductive system DS will compute, from the bottom level upward, for each level of the concept hierarchy, how many distinct values can be generalized from the ground facts in the initial relation. The level, in which the number of distinct values that can be generalized from the initial relation is just smaller than the threshold ATh, becomes the *minimum desirable level* L_A.

Second Pass: Tuple Generalization and Merge. The initial relation is scanned again in this pass and every tuples and its attributes are generalized by DS to the desirable levels computed in the first pass. At the same time, the generalized tuples are compared and identical tuples are merged. The resulting generalized relation becomes the prime relation.

Example 2. The two passes algorithm is applied to the initial relation and the learning task described in Example 1 and the induction is done with respect to the rule-based concept graph described in Example 1 and Fig. 1. The prime relation generated is presented in Table 1. □

Status	Sex	Age	Birthplace	GPA	Vote
undergrad	M	16_25	Canada	average	40
undergrad	M	16_25	Canada	good	20
undergrad	F	16_25	Canada	excellent	10
...	...	...	...	...	...
grad	M	25_30	Canada	poor	6
grad	M	25_30	Canada	good	4
grad	F	25_30	Canada	excellent	4

Table 1. A Prime Relation from the Rule-Based Generalization.

4.2 Induction Phase Two : From Prime Relation to Final Relation

In the second phase, some selected attributes can be generalized further and the generalization-comparison-merge process will be repeated until the size of the relation is smaller than the **generalization relation attribute** RTh. (Some selected attributes can also be removed before the generalization starts.)

In the case of rule-based induction, an **information loss** problem can become an issue in reducing a prime relation to the final relation. In the basic AO induction, attributes in a prime relation can always be generalized further by concept tree ascension because the generalization is based only on the current generalized attribute values. However, this is not the case in the rule-based induction because the application of a rule may depend on the information not currently available in the prime relation. Generalization may sometimes result in *the loss of information*, which could be crucial in the following cases:

(1) A rule may depend on an attribute which has been removed.

(2) A rule may depend on an attribute whose concept level in the prime relation has been generalized too high to match the condition of the rule.

(3) A rule may depend on a condition which can only be evaluated against the initial relation, e.g., the maximum value of certain attribute under certain condition.

For example, one cannot decide whether the first tuple in Table 1, which represents "undergrad with an average GPA", should be generalized according to R_{10} to *weak* or R_{11} to *strong* because the status information (*freshman/sophomore/junior/senior*) has been lost during the previous generalization. In fact, the 40 student tuples which are merged into this tuple may have all the different status.

To tackle the *information loss problem*, a **Backtracking Algorithm** is proposed here. In a generalized relation (intermediate or prime relation), a generalized tuple is the result of merging of a set of tuples in the initial relation. This set of tuples in the initial relation is called the **source set** of the generalized tuple; whereas the generalized tuple is called their **covering tuple**. When the GPA attribute with the values {*poor, average, good, excellent*} in Table 1 are to be further generalized to {*weak, strong*}, more than one rule may be applicable to a tuple in Table 1. For the first tuple in Table 1, both R_{10} and R_{11} may be applicable to different tuples in its source set. Therefore, tuples in its source set may have to be generalized to the two different concepts *weak* and *strong* separately. In order to do this, the covering tuples have to be backtracked to the tuples in their source set.

Principles of the Backtracking Algorithm

1. *Backtrack the tuples in the prime relation to the tuples in their source set.* This can be implemented efficiently by adding a virtual attribute *covering-tuple-id* in the initial relation to record the identity of its corresponding covering tuple.
2. *Further generalize some selected attributes to some higher levels.* This generalization has to be done by the deductive system DS on the the initial relation, not on the prime relation as in the case of Basic AO Induction.
3. *Compare and merge the generalized tuples.* Comparison is performed among the tuples associated with the same covering-tuple-id and relying only on the attributes which are selected for further generalization. Such a comparison and merge of generalized tuples results in an **enhanced-prime relation**.
4. *Map the merged tuples back to the prime relation and split the tuples in the prime relation accordingly.* All the tuples in the enhanced-prime relation sharing the same covering-tuple-id will be mapped to the corresponding covering tuple in the prime relation. Tuples in the prime are then split into several tuples according to this mapping and votes are adjusted. The split tuples are then generalized on the selected attributes to the corresponding values in the enhanced-prime relation.
5. *Merge the generalized tuples in the split prime relation.* If the size of the tuple-merged relation is still larger than the generalization relation threshold, repeat the generalization process until the size of the generalized relation is within the threshold.

Name	Status	Sex	Age	Birthplace	GPA	Covering-tuple-id
–	Junior	M	20	Vancouver	2.3	1
...	...	...	...	...	...	...
–	Sophomore	M	21	Calgary	2.3	1
–	Freshman	M	18	Toronto	2.4	1
...	...	...	...	...	...	...
–	Junior	M	19	Ottawa	3.1	2
–	Ph.D.	M	30	Waterloo	3.9	40

Table 2. Extend the Initial Relation with the attribute Covering-tuple-Id.

Example 3. The Backtracking Algorithm is illustrated by further generalizing the attribute GPA from $\{poor, average, good, excellent\}$ to $\{weak, strong\}$ for the prime relation in Table 1. The initial relation in Example 1, with a virtual attribute *covering-tuple-id* associated, is presented in Table 2. For example, tuples whose *covering-tuple-id* are 1 are those that have been generalized and merged into the first tuple in Table 1. Apply the rules R_9 to R_{13} in Fig. 2 to the initial relation in Table 2 and generalize the GPA attributes to $\{weak, strong\}$. After comparison and merge, the enhanced-prime relation (Table 3) in generated. Since the first two tuples in Table 3 (covering-tuple-id = 1) are mapped to the first tuple in Table 1, the first tuple in the prime relation Table 1 should be split into two. The result of tuple splitting is presented in Table 4. After backtracking and tuple splitting, identical tuples in Table 4 can be merged again to reduce its size. □

Covering-tuple-id	GPA	Vote
1	strong	10
1	weak	30
2	strong	20
...	...	...
40	strong	4

Table 3. Enhanced Prime Relation

Status	Sex	Age	Birthplace	GPA	Vote
undergrad	M	16_25	Canada	strong	10
undergrad	M	16_25	Canada	weak	30
undergrad	M	16_25	Canada	strong	20
...	...	...	...	...	...
grad	F	25_30	Canada	strong	4

Table 4. Result of Splitting the Prime Relation (Table 1)

The Rule-Based Attribute-Oriented Induction process is summarized in the following algorithm:

4.3 Algorithm 1

Rule-Based Attribute-Oriented Induction

```
Input : A learning task specification Ts /* $R$ is the initial relation */
Output: The final relation $R_f$.
Method:
Phase One: Inducing the prime relation $R_p$ from $R$
    Step 1: $R_t := R$; /* $R_t$ is a temporary relation */
    Step 2: for each attribute $A_i$ of $R_t$, compute its minimum desirable level
        $L_i$;
    Step 3: for each tuple in $R_t$ and each attribute $A_i$, deduce its generalized
        value in level $L_i$ and replace the old value of $A_i$ by the deduced value;
        /* merge identical tuples and register the number of tuples merged as
        votes; a virtual attribute covering-tuple-id is added to $R$ to record identity
        of the corresponding covering tuple in $R_t$. */
    Step 4: $R_p := R_t$;
Phase Two: Inducing the final relation $R_f$ from $R_p$ by backtracking
    /* Let N be the size of $R_p$ */
    WHILE N > RTh DO
    BEGIN
    Step 1: Select a set of attributes $A_j$, $(1 \leq j \leq k$, where $k < n)$, from $R_p$ for
        further generalization; for each $A_j$, a desirable level $L_j$ is also decided;
    Step 2: $R_t := R$;
    Step 3: For each tuple in $R_t$ and for each selected attribute $A_j$, deduce its
        generalized value in level $L_j$; /* Compare tuples in $R_t$ by the selected
        attributes and their covering-tuple-id; merge and project the identical tu-
        ples by the selected attributes and the covering-tuple-id into a temporary
        relation $R_{ep}$ (enhanced prime relation). */
    Step 4: $R_{tp} := R_p$;
    Step 5: For each tuple $t$ in $R_{tp}$, split it into a set of tuples corresponding
        to those in $R_{ep}$ whose covering-tuple-id are equal to the identity of $t$;
        /* Merge identical tuples in the split $R_{tp}$. */
    Step 6: $R_p := R_{tp}$;
    END                                                                    □
```

The complexity of the above algorithm can be decomposed into the cost of induction and that of deduction. The induction portion of the algorithm is efficient. However, the cost of the deduction portion will depend on the complexity of the rules and the efficiency of the deductive system DS. The deduction rules in the algorithm are the conditional rules associated with a concept graph, which in most cases, are simple conditional rules. Therefore, it is practical to assume each deduction process is bounded by a constant in the analysis of the algorithm.

Theorem 1. *If the cost of generalizing an attribute to any level is bounded, the complexity of the Rule-Based Attribute-Induction Algorithm is $O(n \log n)$, where n is the number of tuples in the initial data relation.* □

5 Discussion and Conclusion

A rule-based attribute-oriented induction technique is proposed and studied in this paper. The technique extends the previously developed basic attribute-oriented induction method and make the method more generally applicable to databases containing both data and rules. Knowledge discovery in databases has wide applications. First, the knowledge rules discovered are based on a large data set. They represent important regularities in the database. Second, knowledge discovery helps construction of large knowledge-base. The huge numbers of on-line databases with broad applications serve as an enormously rich source for discovering knowledge rules. Third, another important application of knowledge discovery is to support cooperative query answering [6].

This study develops a general model for Rule-Based AO Induction which handles induction on a rule-based concept hierarchy. Even though a rule-based concept graph is substantially more complex than a concept tree, the complexity of the rule-based AO induction remains at the same order to that of the basic induction. Experiments on medium sized data sets have demonstrated the effectiveness and efficiency of the induction method. Our proposed system extends the database learning system to a rule-based induction system. Future experimental work has been planned in this direction. Furthermore, efficient extraction of knowledge from multidatabases, friendly graphical user interface, and efficient knowledge browser are among the interesting topics in knowledge discovery in databases for future research.

References

1. M.L. Brodie and S. Ceri. On Intelligent and Cooperative Information Systems: A Workshop Summary. *International Journal of Intelligent and Cooperative Information Systems*, V1 N2:233–248, 1992.
2. W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. AAAI/MIT Press, 1991.
3. H. Gallaire, J. Minker, and J. Nicolas. Logic and databases: A deductive approach. *ACM Comput. Surv.*, 16:153–185, 1984.
4. J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proc. 18th Int. Conf. Very Large Data Bases*, pages 547–559, Vancouver, Canada, August 1992.
5. J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5:29–40, 1993.
6. A. Motro and Q. Yuan. Querying database knowledge. In *Proc. 1990 ACM-SIGMOD Int. Conf. Management of Data*, pages 173–183, Atlantic City, NJ, June 1990.
7. J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Vols. 1 & 2.* Computer Science Press, 1989.

Case-based Reasoning Applied To A Force Generation Decision Aid

Stephen E. Cross, Advanced Research Projects Agency
Donald F. Roberts, Rome Labs
Alice M. Mulvehill and J. Allen Sears, The MITRE Corporation

1.0 Introduction

This paper describes a knowledge engineering process for developing rich object structures called "force modules" used by military planners, and a tool, ForMAT, that has been developed to support the acquisition, representation, and dissemination of knowledge about military forces for use in solving large scale planning problems. The knowledge acquisition process for force module construction will be tested in late 1994 during exercises where operators are engaged in crisis planning. Current methods for planners and the new KA methodology associated with ForMAT are described in detail in this paper. To date the tool has been focused on acquiring knowledge about the transportation problems and crisis action planning problems associated primarily with the deployment of forces by the Specified Commands and the United States Transportation Command. Knowledge acquired through ForMAT is provided to both researchers and developers who are involved in the ARPA/Rome Laboratory Knowledge Based Planning and Scheduling Initiative (PI), which is described in [1].

2.0 The Problem Domain

ForMAT (Force Management and Analysis Tool) provides an environment in which a force development and planning specialist can view, modify, and create structures called force modules (FM). The FM prescribes a force or set of forces that can be used to satisfy some planning requirement. The FM is typically a grouping of combat, combat support, and combat service support forces, and ranges in size from the smallest combat element to the largest combat element. It may specify accompanying supplies and the required movement resupply and personnel necessary to sustain forces for a minimum of 30 days. The elements of a FM are linked together so that they may be extracted from or adjusted as an entity to enhance the flexibility and usefulness of a plan. In theory, FMs form a library which can be drawn upon to quickly build a new plan.[2]

The United States Transportation Command (USTRANSCOM) is responsible for generating and maintaining the plans by which United States military forces are deployed. This responsibility includes determining the transportation needs for missions short and long, small and very large. For large and very large missions the process by which these transportation plans are constructed can be very complex and time consuming. In addition, transportation plans are constructed to handle "expected" situations, as well as "crisis" situations. Plans built to satisfy "expected" situations are called deliberate plans. Representatives from the various services and commands involved in developing a plan must collectively decide how best to allocate the limited transportation resources (aircraft, ships, trucks and trains) so as to achieve the many military goals of the mission. The end result of this process is an Operational Plan

(OPLAN) which specifies where and when the forces involved in a mission are to be moved. The OPLAN for deliberate planning is stored and maintained until such time as its execution is called for, which may be years after it has been first constructed. At this time the plan will generally have to be modified to fit the particular details of the current situation which often is a crisis situation.

From the perspective of Artificial Intelligence planning, one of the most salient features of these transportation problems is their sheer size. A "small" plan can involve specifying around 10,000 distinct force requirements, each of which must be scheduled. An "average" plan can be twice this size, and large plans are known to exceed 200,000 such separate requirements.

When problems are computationally expensive to solve or when time is a critical factor as in crisis planning, it may be more efficient to provide a mechanism for remembering problems that have been solved previously in order to avoid repeating long computations. This is the basic idea behind case-based reasoning (CBR). CBR systems solve new problems by searching for the best matches of a description of the new problems against descriptions of problems solved in the past. [3]

How past experience is described and how the current problem is mapped into the past is not only a CBR problem but a knowledge acquisition/engineering problem. As a knowledge acquisition/engineering problem, the primary focus is the determination of how a human expert views a problem and how that person determines the similarity between a new problem and one in memory. Knowledge acquisition/engineering can enhance the performance of the CBR system by providing the system with more complete information on how the human expert reasons about past experiences or cases. This type of information directly impacts how cases are indexed and how the similarity between a new problem and a stored case is determined.

In the transportation planning domain, it has been observed that the usage of solutions from the past for application to current similar problems is a technique which is often applied when a planner is trying to determine what type of military force would best satisfy a situation and what that force needs as its support.[4] This matching of situation to force is referred to as force planning, and is fundamental to both deliberate and crisis action planning. Force planning in the deliberate planning situation can involve thousands of elements and is very complicated and subject to frequent revision in response to the changes in the situation. Force planning in the crisis action situation can additionally introduce severe time constraints.[5] Within the military transportation planning domain, this force planning information may be encapsulated in a data structure called the force module (FM).

3.0 Knowledge Acquisition/Requirements Analysis

Knowledge acquisition refers to the initial phases of knowledge-based (or expert) system development. The acquisition process involves eliciting information from experts about their domain. This information is then organized and systematized into an explicit, externalized symbolic model. Later phases of knowledge-based system development recast or embed the externalized model into a computational framework that automates the application of the modeled expertise to solve domain problems.[6]

Knowledge-based systems are typically developed in contexts in which domain specialists are unfamiliar with knowledge acquisition concepts and technologies. A human mediator, a "knowledge engineer", and/or some knowledge engineering tool are used to transition the domain knowledge of the specialist to a formal structure. The process is iterative, time consuming, and continues throughout the lifetime of the system.[7] Success is dependent on many factors: the experience of the knowledge engineer, the continued cooperation of the expert, the stability of the domain, and the robustness of supporting tools. Knowledge acquisition is considered to be a major bottleneck in the development of knowledge-based systems and is generally blamed for impeding widespread commercial application of AI technologies.[8, 9, 10]

The knowledge acquisition process for externalizing expertise tends to result in domain models that are more or less discrepant from the internal models of human knowledge sources [12, 13] Model distortions are introduced throughout the process by the inability of experts to articulate their capabilities or lines of problem-solving reasoning, and through the general problems of communications which result in an imperfect understanding of domain and problem by the knowledge engineer. Many of these problems arise or are aggravated by an ad hoc knowledge acquisition process, i. e., an informal, intuitive approach; when the knowledge engineer is inexperienced; or when the knowledge engineer does not have the full cooperation with the expert.[6]

One way to facilitate knowledge acquisition is to employ a formal methodology. This methodology may specify the use of manual, semi-automated, or fully-automated tools.
Manual techniques, for example can be used to elicit information through retrospective and prospective verbal protocols and questionnaires [13, 14]. More recent methodological developments have concentrated on integrating automated or semi-automated sets of knowledge acquisition tools.[9, 15, 16, 17] Additionally many expert system shells offer some support in the knowledge acquisition process by providing mechanisms for defining classification categories and attributes. Furthermore, certain shells will automatically generate one or more production-like rules based on induction algorithms [18] or other discriminate relationships from this data.

3.1 Knowledge Acquisition/Engineering Using ForMAT

Knowledge engineer in the Planning Initiative focus on the requirements of two user groups: the user or planner who is currently solving transportation planning problems, and the researcher who is developing architectures and representation schemes for reasoning about the planning domain. ForMAT was developed to support the needs of both of these user groups.

ForMAT could be classified as a semi-automated or mixed-initiative knowledge acquisition tool. Through ForMAT, the type of force data that the domain specialists are accustomed to using can be accessed and displayed. Knowledge acquisition sessions to date, have typically consisted of a ForMAT skilled operator, one or more domain specialists, and one or more knowledge engineers. The availability of a skilled ForMAT operator alleviates any requirement for the domain specialists to learn to operate ForMAT, which leaves the domain specialist in a position to dictate how the system should be navigated, what they do not like, and what is required now and for the future. The knowledge engineers present are responsible for manually recording comments and suggestions, as well as any problems experienced in using the system (e.g., speed, failures, errors). They are also responsible for prompting the domain specialists to address certain aspects of the problem domain, thus maintaining some type of overall methodology to the knowledge acquisition session.

ForMAT is not a knowledge acquisition shell. It was derived from a CBR system called SMARTplan.[19] SMARTplan was designed to use CBR to retrieve and modify transportation plans that were built to support military airlift planning. ForMAT uses the CBR aspects of SMARTplan to support knowledge acquisition and engineering. For example, one of the main aspects of knowledge acquisition/engineering is to define an entity (object) in terms of its attributes and values (distinguishing features). The CBR aspect of ForMAT provides an indexing capability that supports this. Another useful knowledge acquisition/engineering activity is to monitor the actions of a user during problem solving, a variant of protocol analysis.[13] ForMATs automated history mechanism maintains a record of each keyboard action that is taken during knowledge acquisition sessions. This history file serves as a kind of protocol record which can then be analyzed by a knowledge engineer in search of procedural information, e.g., "how" force modules are created, modified, and incorporated into a deployment plan. In addition, ForMAT automatically generates attributes and values for the force modules through a series of rules which are applied on the force module data coding techniques.

Through modifications over time ForMAT came to embody the requirements of the domain specialists. It currently contains not only the data required by the domain specialists in order to build and/or maintain a force deployment plan, but additional computational, administrative, and bookkeeping functions that the domain specialists specified as requirements for support in accessing, modifying and maintaining consistency within the existing data as well as in generating new data. In addition, the graphical user interface evolved to satisfy many of the user's specifications. Finally, ForMAT has maintained its ability to communicate (transfer force module information to and from) with currently utilized planning support systems such as DART. Currently ForMAT is viewed by several domain specialists as a force generation and management decision aid.

For the PI researcher, ForMAT provides an environment for transforming the TPFDD-based force module into a set of records within a relational database or into a case within a case memory. ForMAT facilitates this transformation by allowing the contents of a force module to be described as a set of attribute-value pairs. To the CBR researcher, these are the indices to the case; for the database developer, these are the database fields; and for the knowledge engineer, this is a method for describing domain objects.

4.0 Previous Approaches to Knowledge Acquisition/Requirements Analysis

Several semi-automated and fully-automated tools [20] have been developed to support the knowledge engineer in general knowledge acquisition activities, such as acquiring and classifying domain data. Tools range from hypertext-based systems, such as HyperKAT which is a stand-alone, hypertext-based tool that can be used to manage the knowledge acquisition process by providing program documentation, knowledge base traceability, and instruction on knowledge acquisition techniques [21]; to inductive-based systems, such as Scotty [22] which generates rules automatically from examples. In addition, an entire set of automated knowledge acquisition tools have been built on the theoretical foundation of personal construct theory which employ repertory grids by which people, objects, situations, or other kinds of elements are either categorized, rated, or rank-ordered on a set of constructs.[23, 24]

Using an automated knowledge acquisition shell can be quite effective: the knowledge engineer's knowledge acquisition role is effectively replaced in the development cycle by tools that automate the elicitation of knowledge from domain experts. The problem of transferring expertise in AI technologies per se is finessed by providing a turnkey "black box" expert system generating tool. However, single individuals are rarely masters of both domain expertise and AI technologies. When a domain expert is interacting with a knowledge engineering tool (such as an expert system development shell), accurate representation of the domain and of problem solving strategies within the domain is dependent upon the expert's grasp of the intricacies of the development shell. [6] Unfortunately, as tools become more sophisticated, they generally become more difficult to comprehend and use.[15]

5.0 Case-Based Reasoning

Case-based reasoning (CBR) is a machine learning approach which utilizes past experience to solve current problems. The CBR approach solves new problems by retrieving previous similar problems and adapting their solutions to the current problem. The resulting computation is proportional to the magnitude of the difference between the new problem and the retrieved case, rather than the complexity of the problems themselves.[19]

A key issue in using CBR to solve problems is locating the previous cases which are relevant to the new problem. This is generally accomplished by choosing a set of features to describe both the new problem and the previous cases. The case whose feature vector is most similar to that of the new problem is retrieved. Typically, cases are represented as a set of attribute-value pairs describing the case. The success of CBR depends upon the judicious choice of these attributes. Some attributes are easily determined, being primitive features describing the problem. For the planning domain, these might include the type of mission and the geographical location of the mission. In such problems, however, relying on primitive features is usually not practical. The large number of features needed to describe complex problems results in inefficient retrieval. More importantly, primitive features often do not capture the high level relationships required for generalization. For example, the high level description "useful in conditions of poor visibility" is more informative to a user than the combination of primitive features on which it is based.

The alternative is to rely on abstract features which capture only those high level characteristics of a problem that are relevant to the particular task. These features may be determined in advance and built into the system. However, considerable knowledge engineering is required to arrive at a set of features that correctly and completely characterize the domain. Furthermore, the tasks encountered may vary over time or from one user to the next, resulting in features being required that were not anticipated during the knowledge engineering process.

In ForMAT, feature (attribute-value) construction is performed in two ways: (a) new terms, are used to index cases, and are acquired through interaction with the user; and (b) a set of domain specific rules are employed to transform information implicit in force module unit record data into explicit features (attribute-value pairs). In the latter method, for example, if the "svc" field of a unit record is "a", then rule XA would get fired to generate a feature called "service" with the value "army" and append this to the FM.

User interaction is also used to acquire the data missing from the TPFDD's explicit declaration, but often hidden in a FM description field (e.g., descriptions of the "capabilities" of the FM, and of why it was built, e.g., the situation; and of its limitations and constraints). To explicitly index this information with the FM, the user is asked to define attributes with associated values.

Conceivably, there is no limit to the number of attributes and values that can be used to describe something in ForMAT, and there is currently no method employed to track the use of synonyms. However, replication of exact or similar features is inhibited by providing the user with a list of all defined features, and for each feature, a list of all of its defined values. Additionally, statistics are generated on the frequency by which attribute-value pairs are used to describe some number of cases. Attribute-value pairs with very low frequency rates are manually inspected to determine if there is some other attribute-value pair being used instead or if the attribute-value pair offers no value as a index. When this is the case, the attribute and/or value is eliminated. Interestingly, our reviews of attribute-value usage has indicated that some indices that were specified to be of value by the domain specialists have, in fact, never been employed to index a force module.

5.1 The Set-Theoretic Case Memory

ForMAT stores FMs as cases in a set-theoretic case memory. The set-theoretic case memory is organized as tables of attribute-value pairs For each attribute-value key, the set of all FMs which match that attribute-value pair (whether exactly or by abstraction) is stored in the table. Retrieval is expressed as a query on the attribute-value pairs, e.g., (service = army) ^ (capability = anti-tank). Retrieval is performed by executing increasingly generalized queries until some force module(s) are returned.

The set-theoretic case memory has several advantages: (a) different similarity metrics may be used without re-organizing the case memory, which is important because a planner's changing needs may necessitate using different similarity metrics; (b) value hierarchies can be changed and new features can be defined without significant changes to the memory organization, and this is especially important with a dynamic representation that will likely change quickly early in the knowledge acquisition process; (c) retrievals can be flexible, e.g., conjunction, disjunction, negation, use of wildcards, etc.; and (d) certain types of retrievals can be performed very quickly, i.e., if there are FMs close to, or within, the subspace identified by a particular query with a given similarity metric, they will be found quickly.

6.0 System Description And Status

ForMAT draws upon our research experience in CBR and our empirical background in knowledge acquisition. The tool was derived from earlier CBR work on a transportation planning based system called SMARTplan. ForMAT provides an environment for creating, modifying, indexing, and analyzing one specific CBR aspect of transportation planning, namely the usage of force modules (FMs). Methods for updating FMs, storing hierarchies, etc., has been implemented. Menus have been tailored to allow planners to load TPFDDs, load/save case bases, and load/save domains, while increasing ForMAT's flexibility in handling multiple knowledge bases. ForMAT currently contains several hundred force module cases.

7.0 Future Plans

ForMAT was designed to support the acquisition of domain knowledge pertaining to the development and management of force modules by force development and planning specialists. In the acquisition of knowledge about that domain problem, ForMAT has evolved into a Force Module Analysis and Management Tool. Its interface design/layout and it's functional capabilities have significantly been driven by the requirements and recommendations of the domain specialists and operational users. Due to acceptance of initial capability by the user community, ForMAT has been selected to be enhanced and then evaluated in terns of operational user requirements.

In the last half of 1994 ForMAT will be integrated with other planning tools and then demonstrated in a crisis planning situation as part of a concept development phase for distributed collaborative planning. The evaluation exercise will focus on Joint Service operations, and will provide a testing ground for new capabilities in multi-mission planning and execution support. ForMAT will be used as an adaptive Force Package editing tool, and to support Service Components in force generation and selection. During the 1994 evaluation most hands-on use will be by technical personnel who are helping operators use the equipment and software tools.

The products from ForMAT during the demonstration will be both TPFDDs and specification of assigned forces used to satisfy specific and implied tasks. Time and quality of planning will be the metrics for effectiveness: for example the time taken to produce these products, and the quality of plan selection following the semi-automated force generation process. User satisfaction compared to current methods of operation will also be evaluated. It is expected that mid-term exams will show that distributed collaborative planning reduces the time it takes to generate plans, while simultaneously improving planning quality by providing a broader range of plans to be considered. The use of ForMAT in force generation and editing of force packages is expected to directly contribute to the improved planning performance. If these expectations are realized then the planning system with the ForMAT component will be engineered to make them readily usable by actual operational personnel. Final exams will then be conducted in 1995 in order to evaluated operational impact of the new tools and the distributed collaborative planning infrastructure.

This process of conducting user-centered development and evaluations utilizing intermediate feasibility demonstrations (IFDs) is becoming a repeatable way to build systems quickly that meet the real needs of operational users. It has been shown that user-centered software engineering (USE) produces a new class of systems that do "make a significant difference" in operational performance leading to reduced costs and improved readiness. Moreover, the tools and decision aids will be engineered well enough to be left in place for the user. ForMAT is yet another example of a scalable intelligent systems tool that benefits from the USE process. ForMAT has been shown to be interoperable with other emerging intelligent systems tools, and will provide an important addition to the growing number of capabilities aimed at improving crisis management planning and execution for the Department of Defense and the Federal Government.

8.0 Summary

This paper describes knowledge acquisition methods and a force generation decision aid, ForMAT, that has been developed to support the acquisition, representation, and dissemination of knowledge about military forces for use in solving large scale planning problems. The paper describes the problem domain, how the knowledge acquisition was performed in general and with the use of ForMAT through a coupling of knowledge acquisition and case-based reasoning techniques. The paper concludes with a description of the status of the system and a description of how the system will be evaluated for performance from a user-centered perspective in the context of a complex planning scenario.

Bibliography

[1] Walker, E. and Cross, S. (Eds), Proceedings of the ARPA-Rome Laboratory Planning Initiative Annual Workshop, to be published, Morgan-Kaufmann Publishers, June 1994.

[2] Joint Pub 5-03.1, Joint Operations Planning System.

[3] Kolodner, J. L., and C. K. Riesbeck, "Case-based Reasoning Tutorial", International Joint Conference on Artificial Intelligence, August 1989.

[4] Mulvehill, A. M., and Christey, S. M., "Acquiring and Representing Planning Knowledge Year End Report", MITRE MTR-92B0000146, October 1992, The MITRE Corporation, Bedford, MA 01730.

[5] Magnes, J. E., Lt. Col. USAF, The Joint Staff Officer's Guide, 1991.

[6] Mulvehill, A. M. and, Adler, R. M., "On the Use of Manual, Semi-Automated and Fully-Automated Knowledge Engineering Tools", The MITRE Corporation, MITRE M-91-9, February 1991, Bedford, MA 10730.

[7] Siemens, R. W., Golden, M., Ferguson, J. C., StarPlan II: Evolution of an Expert System, Proceedings of AAAI-86, August 11 - 15, 1986, Philadelphia, PA.

[8] Hart, A., Knowledge Acquisition for Expert Systems, McGraw-Hill, New York; 1986.

[9] Marcus, S. (ed), Automating Knowledge Acquisition for Expert Systems, Kluwer Academic Publishers, 1988.

[10] Prerau, D. S., "Knowledge Acquisition in the Development of a Large Expert System", Summer 1987, AI Magazine.

[11] Collins, A., "Component Models of Physical Systems", The Seventh Annual Conference of The Cognitive Science Society, Irvine, CA 1985.

[12] Gentner, D., and Stevens, A. L. (eds), Mental Models, Hillsdale, NJ: Erlbaum, 1983.

[13] Ericsson K. A., and Simon H. A., Protocol Analysis, MIT Press, 1984.

[14] Hayes-Roth, F., Waterman, D. A., and Lenat, D. B., Building Expert Systems, Addison-Wesley Publishing Company, Inc., Massachusetts, 1983.

[15] Boose, J. H., and Bradshaw, J. M., "NEONETS: Capturing Expert System Knowledge in Hierarchical Rating Grids", IEEE, 1986.

[16] Rothenberg J., Paul, J., Kameny, I., Kipps, J. R., and Swensor, M., "Evaluating Expert System Tools", R-3542-DARPA, July 1987.

[17] Sowa, J. F., Conceptual Structures, Addison-Wesley, Massachusetts, 1984.

[18] Quinlan, J. R., "Learning Efficient Classification Procedures and their Application to Chess End Games", Machine Learning: An Artificial Intelligence Approach, ed. by Michalski, R., Carbonell, J., and Mitchell, T., Morgan Kaufmann Publishers, Los Altos CA, 1983.

[19] Coley, S. M., Connolly, D., Koton, P. K., McAlpin, S., and Mulvehill, A. M., "SMARTPLAN: A Case Based Resource Allocation and Scheduling System, Final Report", MITRE MTR-11270, The MITRE Corporation, Bedford, MA 01730.

[20] McGraw, K. L, and Westphal, C. R., (eds), Readings in Knowledge Acquisition, Ellis Horwood, New York, 1990.

[21] McGraw, K. L., "HyperKAT: A Tool To Manage and Document Knowledge Acquisition", from: McGraw, K. L, and Westphal, C. R., (eds), Readings in Knowledge Acquisition, Ellis Horwood, New York, 1990.

[22] Modesitt, K. L., "Inductive Knowledge Acquisition: A Case Study of Scotty", from McGraw, K. L, and Westphal, C. R. (eds), Readings in Knowledge Acquisition, Ellis Horwood, New York, 1990

[23] Boose, J. H., "Personal Construct Theory and the Transfer of Human Expertise", Proceedings of the National Conference on Artificial Intelligence, Austin Texas, 1984.

[24] Ford, K. M., Adams-Webber, J. R., Stahl, H. A., and Bringmann, M. W., "Constructivist Approaches to Automated Knowledge Acquisition", , from McGraw, K. L, and Westphal, C. R. (eds), Readings in Knowledge Acquisition, Ellis Horwood, New York, 1990

A Case-Based Reasoning Approach for Associative Query Answering*

Gilles Fouqué, Wesley W. Chu, and Henrick Yau

Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
{gfouque, wwc, henrick}@cs.ucla.edu

Abstract. Traditional DBMS only retrieves data that perfectly match the user query and also requires the user to know the detailed database schema. Often, it is desirable to obtain additional relevant information to a query. In this paper, we present a method to provide useful information to the user that he does not explicitly asked for. Such domain specific knowledge associated to a given query depends on each user's goal and knowledge. Thus, we propose using the Case Based Reasoning paradigm to integrate past user experience and the current goal in order to guide the association. Useful associations are incrementally acquired from observations of past experiences and adapted to answer the current user query. A prototype of the associative query answering system using the proposed method has been implemented on top of the cooperative database system, CoBase, at UCLA. Our preliminary experimental results reveal that it is a feasible and scalable method for association control.

1 Introduction

Traditional query answering systems require a perfect match between the user query and the data to be retrieved. This approach requires the user to have a detailed knowledge of both the database schema and the query language. Moreover, the user has to know exactly what information to search. *Cooperative query answering* has been recently developed to relax these limitations where different approaches are used to broaden the scope of the user query. Cooperative databases extend the traditional database query languages by providing new operators (*"near-to"*, *"similar-to"*) and by providing explanation of the relaxation process and quality of the answers [CMB93]. In this paper, we focus on a specific aspect of cooperative databases: the inclusion of information not explicitly asked by the user but relevant to his query. This operation is called *associative query answering* [CD91][CHF92] that provides additional information that either the user does not know how to ask for or does not know is available.

We propose to use query context to infer the user search goal. The query context is based on the analysis of the user's previous experience: his previous

* This work is supported by DARPA contract N00174-91-C-0107

searches, and the ability of the system to infer his successive goals. A Case-Based Reasoner[RS89] is used to store, index, select, and adapt this experience and while learning from previous system successes and failures[FM93].

In this paper, we first present an example of associative query answering, followed by a presentation of the association system architecture. Then, we discuss CBR learning from user feedback. Finally, we present our CBR implementation.

2 Example of Associative Query Answering

Let us consider the following query: *"List all airports in Tunisia with runway length approximately equal to 10.000 feet"*

The answers provided by CoBase[CMB93] with approximate operators are:

Airport Name	*Runway Length in Feet*
Jerba	10171.00
Monastir	9700.00
Tunis	10500.00

It would be desirable to provide additional relevant information, even when not requested in the query. Further, this additional information depends on user type. For example, in the context of a transportation application, if the user is a pilot, then information on weather and runway quality are useful as follows:

Query Answers		Associative Query Answers	
Airport Name	*Runway Length in Feet*	*Weather Condition*	*Runway Quality*
Jerba	10171.00	SUNNY	GOOD
Monastir	9700.00	RAIN	GOOD
Tunis	10500.00	FOGGY	DAMAGED

If the user is a transportation planner trying to find the best schedule for delivery, then relevant information for the user might be: the category of the airport (civilian or military), the availability of a close seaport and the availability of refrigeration capacities as shown below:

Query Answers		Associative Query Answers		
Airport Name	*Runway Length in Feet*	*Military or Civilian*	*Seaport Name*	*Refrigerated Storage Capacity in Tons*
Jerba	10171.00	C	None	0
Monastir	9700.00	C	Monastir	0
Tunis	10500.00	C	Tunis	1000.00

Therefore, in selecting the list of relevant subjects, the associations are based on query context as well as user type.

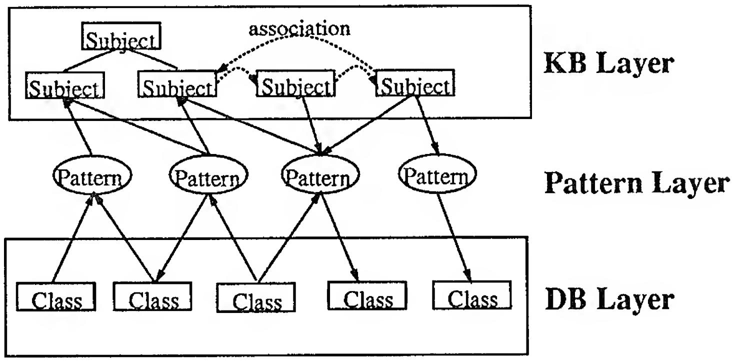


Fig. 1. *The DB-pattern-KB Architecture*

3 The association facility

3.1 The DB-pattern-KB architecture

Associative query answering requires the linkage of data and knowledge. We propose a DB-pattern-KB architecture (Figure *1*) for associative query answering.

- *The DataBase (DB) layer:* contains the relational or object-oriented databases. A class encapsulates the properties common to a group of data.
- *The Knowledge Base (KB) layer:* contains the application domain knowledge not already included in the database layer. This knowledge is composed of rules and facts. The KB layer is organized into sets corresponding to subjects as defined by the domain expert. The associations are performed in the KB layer by explicit links between different subjects.
- *The Pattern layer:* provides the interface between the DB and the KB layers. A pattern is a set of query conditions belonging to one or several subjects. It provides a precise and applicable knowledge representation.

Associative query answering is accomplished through tracing the dependencies between data distributed in multiple subjects. In such a framework, the knowledge base and database layers can be maintained independently[CC92].

3.2 Association control

Since association is a transitive process depending on user type and goal, it needs control for selection termination. In the previous example, we assumed that such association control was available. Otherwise, all the related subjects in the DB-pattern-KB framework would be selected for association. Therefore, we propose to use Case Based Reasoning (CBR) to control the association and to acquire information about the user goals. The user queries and their answers are grouped into cases. The Case Memory is composed of previous cases related by links between a case and its associations. Weights are assigned to the links to represent their usefulness for the association to a specific query context and user type. The CBR uses user's feedback to dynamically update the link weight between a case and its associations as its learning from its successes and failures.

3.3 The association representation

A set of queries with their corresponding answers and a domain specific knowledge base are stored as cases in the Case Memory. The cases provide attributes for association and their usefulness for association is learned during the use of the CBR. Associations to a user query are achieved by first adding a set of attributes to a select list of the user's query and then running the modified query.

In the Case Memory, an association is represented by an *association link* between two queries (cases) that share one or more attributes. Because all the attributes are not equally important for an association, a weight is assigned to each attribute of an association link. The larger the weight, the more relevant the association.

The cases in the Case Memory are acquired incrementally during the use of the CBR. There are two approaches to acquire the initial set of cases: *(a)* an automatic generation of queries based on the distribution of relations and attributes in the database, or *(b)* based on user's previous transactions.

In the first case, the association weights are initialized according to the uniform distribution. Thus, prior to any learning, each attribute has an identical chance for association. The association initialization weight for an attribute a_k shared by two cases C_i and C_j is:

$$W'(C_i \rightarrow C_j, a_k) = 1/t, \text{ for } k = 1, \ldots, t \tag{1}$$

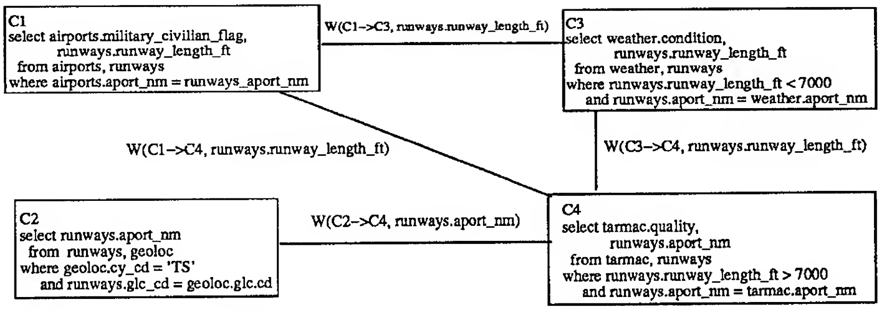
When previous user transactions are used for the initialization of the association weights, the initialization weight of an association link can be based on additional information related to the distribution of each attribute in the Case Memory:

$$W(C_i \rightarrow C_j, a_k) = -\log_2(F(a_k)) \tag{2}$$

where: $\sum_{k=1}^{t} F(a_k) = 1$;
$F(a_k)$ = frequency of occurrence of attribute a_k in all the Case Memory.

Since a seldomly used attribute carries more specific information for association, its weight is initialized with a higher value than that of frequently used attributes *(2)*. A Case Memory example is shown in Figure *2*. C_1, C_2, C_3, and C_4 are four previous user queries. The initialization of association weights, $W(C_i \rightarrow C_j, a_k)$, is symmetric and equal to $W(C_j \rightarrow C_i, a_k)$, for $i, j = 1, 2, 3, 4; i \neq j$. The Case Memory includes 4 association links, each one with a single attribute: *runway_length_ft* or *airport_name*. Thus, the total number of attributes used in the association links is $4 * 2 = 8$. From *(2)*, the initial association weights for attribute *runway_length_ft* and *airport_name* are[2]: $W(C_1 \rightarrow C_3, runway\_length\_ft) =$

[2] latter, the association weights are normalized by the highest association weight in the Case Memory



```
C1
select airports.military_civilian_flag,
         runways.runway_length_ft
  from airports, runways
where airports.aport_nm = runways_aport_nm
```

```
C3
select weather.condition,
         runways.runway_length_ft
  from weather, runways
where runways.runway_length_ft < 7000
    and runways.aport_nm = weather.aport_nm
```

```
C2
select runways.aport_nm
  from  runways, geoloc
where geoloc.cy_cd = 'TS'
    and runways.glc_cd = geoloc.glc.cd
```

```
C4
select tarmac.quality,
         runways.aport_nm
  from tarmac, runways
where runways.runway_length_ft > 7000
    and runways.aport_nm = tarmac.aport_nm
```

W(C1->C3, runways.runway_length_ft)
W(C1->C4, runways.runway_length_ft)
W(C3->C4, runways.runway_length_ft)
W(C2->C4, runways.aport_nm)

Fig. 2. *Case Memory*

$W(C_1 \rightarrow C_4, \mathit{runway\_length\_ft}) = W(C_3 \rightarrow C_4, \mathit{runway\_length\_ft}) = -\log_2(F(\mathit{runway\_length\_ft})) = -\log_2(3 * 2/8) = 0.42$ and $W(C_2 \rightarrow C_4, \mathit{airport\_name}) = -\log_2(F(\mathit{airport\_name})) = -\log_2(2 * 1/8) = 2$.

The attributes associated to the attribute *airport-name* is considered more specific than attributes associated to *runway_length_ft*. Consequently, if the user query includes both attributes, the attributes provided for association by the association link through *airport_name* between C_2 and C_4 are more important than those provided by the association links *runway_length_ft* between C_1 and C_3, between C_1 and C_4, and between C_3 and C_4.

3.4 Selection and ranking of associated attributes

We shall now discuss the selection and ranking processes of finding the set of similar cases for the user query. To find the attributes associated with the user query Q, the CBR searches the Case Memory for all association links contained in the set of attributes in Q.

The CBR searches in the Case Memory for the association links that use one or more of the attributes in Q. The cases C_i's related to these association links are selected. The CBR evaluates their usefulness from the comparison of Q and C_i's query contexts, and the weights of the association links.

Two queries perfectly match when they have identical relations, attributes, and attribute values. When two queries share the same relation and attribute but with different values, we use domain specific knowledge to measure the relevance of the substitution of one value by another. This knowledge is represented in the Type Abstraction Hierarchies (TAH) where attribute values are clustered depending on a nearness function represented as a correlation measure[MC93]. This nearness function is based on several features: *a) answer set locality* — instances that are often selected together are considered to have higher nearness; *b) inter-attribute relationship* — the nearness of two values of the same attribute depends on the number of values they share with the other attributes in the relation. Therefore, two cities both having a seaport and in the same country are more correlated than two cities having no shared attribute values. Thus, two queries with identical attributes but poorly correlated values are not equivalent.

The similarity between the user query Q and the query context of case C_i is:

$$S(Q, C_i) = \sum_{k=1}^{t} C_i(a_k) * Q(a_k) * \gamma(a_k; x, y) \tag{3}$$

where:

$Q(a_k) = \begin{cases} 1 \text{ if the attribute } a_k \text{ is used in the user query } Q, \text{ for } k = 1, \ldots, t, \\ 0 \text{ otherwise} \end{cases}$

t = total number of attributes in the database.

$C_i(a_k) = \begin{cases} 1 \text{ if the attribute } a_k \text{ is used in the case } C_i, \text{ for } k = 1, \ldots, t \\ 0 \text{ otherwise} \end{cases}$

$\gamma(a_k; x, y)$ = the correlation value between the values x in Q and y in C_i in the TAH for attribute a_k.

The usefulness for Q of an attribute a in case C_j associated with C_i, $U(Q, a)$, depends on the similarity between the user query Q and C_i, $S(Q, C_i)$, and the weight of the attributes in the association link, $W(C_i \rightarrow C_j)$; that is,

$$U(Q, a) = S(Q, C_i) * \sum_{k=1, a \neq a_k}^{t} W(C_i \rightarrow C_j, a_k) \tag{4}$$

The list of associated attributes is ranked based on their usefulness for association to Q *(4)* and presented to the user for feedback. The selected attributes are appended to the user query to derive the associated information.

3.5 An indexing schema for reducing computation complexity

The user query Q is first matched with the association links to select the cases that are likely to provide useful associated attributes. A traditional browsing of the Case Memory association links is computationally too costly since it increases exponentially with the number of stored cases. To reduce the computation complexity, we propose to use an inverted file of the Case Memory to compare Q to a limited set of association links that are more likely to lead to useful cases for associations. The inverted file can be generated immediately after the initialization of the Case Memory and updated during the learning process. In the inverted file a branch is created in the indexing tree for each association link in the Case Memory. The attributes in a branch are ordered based on their weights in the association links. The total weight of each branch is computed by summing the corresponding association weights of that branch. The highest the total weight, the more relevant is the associated attribute provided by that association link. A link is created between the inverted file and the Case Memory to list all the association links for each branch leaf.

The selection process using the proposed indexing schema is as follows: *a)* based on the set of possible combinations of attributes used in user query Q,

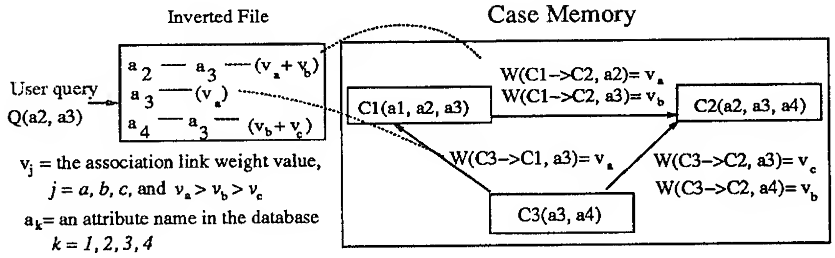


Fig. 3. *The indexing schema*

select a set of association links and their corresponding total weight from the indexing trees in the inverted file, *b)* select the m highest association links and their corresponding cases, *c)* compute the usefulness of association for the attributes of each association link from *(4)*.

By using this inverted file, the complexity of the selection process is in $O(n^2)$ where n is the number of attributes in Q. Note the complexity is independent of the number of stored cases. This efficient indexing schema is possible because we know the entire set of database attributes used in the user queries.

An example of the indexing schema with three cases $C_1(a_1, a_2, a_3), C_2(a_2, a_3, a_4)$, and $C_3(a_3, a_4)$ is presented in Figure *3*. Three association links are represented in the Case Memory with the attributes shared by the related cases and weights. For the branches included in the user query Q, the relations between the leaves of the inverted file trees and their corresponding association links are represented by broken lines. Q has two attributes a_2 and a_3, four combinations of these attributes are possible: $\{(a_2), (a_3), (a_2, a_3), (a_3, a_2)\}$. Two of these are in the indexing trees: (a_3) with the weight v_a, and (a_2, a_3) with the weight $v_a + v_b$. Cases C_1 and C_3 are selected for association with Q. Since the association weight for C_1 is larger than that of C_3, C_1 is appended to Q before C_3. A threshold can be used based on a maximum number of association links or on a minimum association weight to limit the number of associated attributes.

4 Learning from user feedback

User feedback is crucial for systems to understand the difference between the user query and the user information needs [SM83]. The user selects the associated attributes based on its relevancy to his problem. The unselected associated attributes are considered irrelevant. Such user feedback is used by the CBR to update the association weights. The CBR estimates the average association weight for an associated attribute a from the weight of the attributes a_k $(k = 1, \ldots, t)$ from all the cases that participated in selecting attribute a as follows:

$$W(a_k; a) = \frac{W(C_i \rightarrow C_j, a_k; a) + \ldots + W(C_m \rightarrow C_n, a_k; a)}{N} \qquad (5)$$

where: $W(a_k; a)$ = average association weight of a_k for association with attribute a,
$W(C_i \rightarrow C_j, a_k; a)$ = association weight of a_k in the association link from C_i to C_j that participated in selecting attribute a,
a_k = attribute used in selecting attribute a,
N= number of association links that use a_k in selecting the attribute a.

The association weights are updated after the selection of associated attributes. In order for the CBR to appropriately update the association weights, a confidence measure $\alpha(a,p)$, $0 \leq \alpha(a,p) \leq 1$, is used to represent the confidence in the current user feedback for association with attribute a, based on the p previous associations on a. One possible approach is to use Bayesian statistics to estimate $\alpha(a,p)$.

After the $n+1^{th}$ user feedback for an attribute a_k that participates in selecting associated attribute a, the association weight, $W_{n+1}(C_i \rightarrow C_j, a_k; a)$, is the sum of the association weight at the n^{th} user feedback and the weight contribution from the current feedback.

$$W_{n+1}(C_i \rightarrow C_j, a_k; a) = W_n(C_i \rightarrow C_j, a_k; a) + f(W_n(C_i \rightarrow C_j, a_k; a), \alpha(a,p), W(a_k; a)) \quad (6)$$

where: $W_n(C_i \rightarrow C_j, a_k; a)$ = association weight after n feedbacks for the link between C_i and C_j for a_k to associate with a,
$W_{n+1}(C_i \rightarrow C_j, a_k; a)$ = association weight after $n+1$ feedbacks for the link between C_i and C_j for a_k to associate with a,
f = a learning function.

Similarly, the association weights updating formula for irrelevant associated attributes is:

$$W_{n+1}(C_i \rightarrow C_j, a_k; a) = W_n(C_i \rightarrow C_j, a_k; a) - f(W_n(C_i \rightarrow C_j, a_k; a), \alpha(a,p), W(a_k; a)) \quad (7)$$

Equations *(6)* and *(7)* are adapted from IR formulas in the vector processing environment to modify the initial query from user feedback[SB90].

The choice of the learning function depends on the application, the number of cases in the Case Memory, the attribute distribution in the Case Memory, the desired accuracy, and the efficiency of associations. A good learning function should stabilize the Case Memory after a certain training period. We have tested the user feedback mechanism with the following linear learning function:

$$W_{n+1}(C_i \rightarrow C_j, a_k; a) = W_n(C_i \rightarrow C_j, a_k; a) + (1 - W_n(C_i \rightarrow C_j, a_k; a)) * W(a_k; a) \quad (8)$$

Our experience with this simple learning function reveals that it provides a fast learning from initial user feedbacks. The learning slows down as more relevant feedbacks are provided. However, this learning function is very sensitive to irrelevant feedbacks. Therefore, more learning functions need to be developed. We plan to experiment and classify a selected set of learning functions (e.g. polynomials) to study the system behavior depending on user feedback quality.

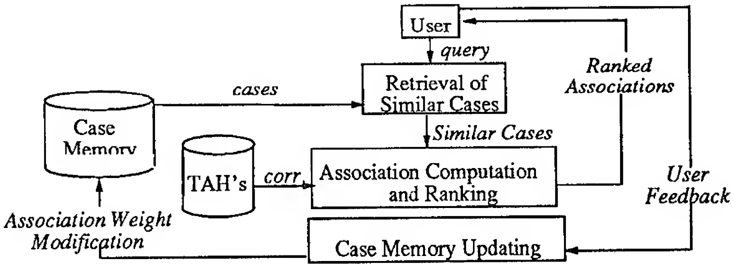


Fig. 4. *Associative Query Answering Algorithm*

To keep the size of the Case Memory manageable, certain links are discarded when their association weight decreases. One approach to avoid repetitive creation/deletion of links is to keep the weak association links in the Case Memory and change their status from active to passive. Active cases are used during the selection process, and the passive cases are are kept in the Case Memory as irrelevant cases. Updates of the Case Memory are applied to all cases.

5 Implementation

The association facility has been implemented on top of the cooperative database system CoBase[CMB93]. A predefined set of generic user models[MP92] is used. The association algorithm can be divided into three processes (Figure *4*):

- *Selection of similar cases*: query conditions are extracted as indexing features from the user query by its *select* and *where* clauses. The indexing schema is used to select association links included in the user query, while the set of cases similar to the user query selected from these association links.
- *Selection and ranking of associated attributes*: the similarity between the attribute values of the selected similar cases and the user query is computed from their correlation in pairwise TAH's for symbolic attributes[MC93] or from their relative difference for numeric attributes[CC94]. Associations are extracted from new attributes of the cases associated with the set of similar cases. Based on the user type, irrelevant associated attributes are filtered out. The selected cases for association are ranked by their usefulness*(4)*.
- *Updating of association weights*: the ranked set of associated attributes is presented to the user and his feedbacks are recorded. The association weights are then updated based on user feedbacks.

Currently, our Case Memory testbed consists of about 400 cases and runs on a SUN Sparc 10. Our experiments reveal that the time required to select and rank the four best attributes (a pre-specified association size threshold) for a given user query is less than 2 seconds, which is less than the query processing time.

6 Conclusion

In this paper we have developed a measure of "usefulness of association" to select relevant attributes for association. The association control is based on experience acquisition, user model, and query context. The evolution of association requirements are managed via CBR which adapts its knowledge from user feedbacks. The association facility infers the user goal of the current transaction from his previous transactions.
We have constructed a prototype on top of CoBase at UCLA for generating associate answers that use CBR for association control. Our preliminary experiences reveal that such association control is both feasible and scalable. Further experimentation is under way to study the behavior of the learning algorithm of using user feedback to modify association weights for selected applications.

References

[CC92] W.W. Chu and Q. Chen. Neighborhood and Associative Query Answering. *Intelligent Information Systems*, 1(3-4):355–382, 1992.

[CC94] W. Chu and K. Chiang. Abstraction of High Level Concepts from Numerical Values in Databases. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Seattle, 1994.

[CD91] F. Cuppens and R. Demolombe. Extending Answer to Neighbor Entities in a Cooperative Answering Context. *Decision Support System*, pages 1–11, 1991.

[CHF92] M.A. Casanova, A.S. Hemerly, and A.L. Furtado. A Declarative Conceptual Modelling Language: Description and Example Applications. In *Proceedings of 4th International Conference on Advanced Information Systems Engineering*, pages 589–611, Manchester, UK, 1992.

[CMB93] W.W. Chu, M. Merzbacher, and L. Berkovich. The Design and Implementation of CoBase. In *Proceedings of SIGMOD 93*, pages 517–522, May 1993.

[FM93] G. Fouqué and S. Matwin. A Case-Based Approach to Software Reuse. *Journal of Intelligent Information Systems*, 2(2):165–197, 1993.

[MC93] M. Merzbacher and W.W. Chu. Pattern-based clustering for database attribute values. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, Washington, D.C., 1993.

[MP92] J.D. Moore and C.L. Paris. Exploiting User Feedback to Compensate for the Unreliability of User Models. *User Modeling and User-Adapted Interaction*, 2(4):287–330, 1992.

[RS89] C.K. Riesbeck and R.S. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, N.J., 1989.

[SB90] G. Salton and C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–297, June 1990.

[SM83] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* NcGraaw-Hill, New York, 1983.

Efficient Execution of Recursive Queries Through Controlled Binding Propagation*

Sergio Greco
DEIS Dept.
Univ. of Calabria
87030 Rende, Italy
greco@ccuscl.unical.it

Carlo Zaniolo
Computer Science Dept.
Univ. of California
Los Angeles, CA 90024, USA
zaniolo@cs.ucla.edu

Abstract

This paper presents a new method for the computation of bound linear Datalog queries and compares its performance to that of other bottom-up execution strategies. The technique, called pushdown *method, uses virtual stacks to store information on the current state of the evaluation thus ensuring both termination and efficient execution. The asymptotic worst-case behavior and experimental results for the new method compare favorably against those of previous methods.*

1 Introduction

Several strategies has been proposed for the optimization of bound Datalog queries. Most of these strategies rewrite the original program into a query-equivalent new program which can be evaluated more efficiently by the semi-naive algorithm [2, 14, 4, 3, 9, 10, 12]. The improved efficiency of the rewritten program is due to the fact that it restricts the search to the portion of the underlying database that is relevant to the query. Other techniques present special algorithms that compute directly the original program [7, 17, 1, 6, 18].

Rewriting-based techniques can be classified according to their generality. Techniques that can be applied to all programs include the *magic-set* method and the *supplementary magic-set* method [2, 4]. Specialized techniques include the *factorization* and reduction of programs [9], the combination of the propagation of bindings with a successive reduction [10], and the *counting* method [2, 13, 1, 6]. These specialized techniques are important, since many programs of practical interest contain only linear recursive rules, to which these techniques apply, yielding an order of magnitude improvement in efficiency [16]. Comparisons between the magic-set method and the counting method can be found in [3, 8].

In this paper, we present a unifying framework for the efficient implementation of linear rules using a method, called *pushdown* method, which is based on the implicit use of stacks. For counting queries stacks reduce to counters and thus the method reduces to the classical counting method; for left-, right-, and mixed-linear queries the stacks can be deleted and the method reduces to the left-, right-, and mixed-linear method [5].

*The work of the first author has been supported by the CNR project "Sistemi Informatici e Calcolo Parallelo" and by the MURST project "Metodi Formali per Basi di Dati".

2 Preliminaries

We now recall some basic concepts [16]. Predicates whose definition consists of only ground facts are called *base* predicates while all other predicates are called *derived.* The set of facts whose predicate symbol is a base predicate defines the *database* while the set of clauses whose head predicate symbols is a derived predicate symbol defines the *program.* A *query* is a pair (G, P) where G is a predicate called *query-goal* and P is a program. The *answer* to a query (G, P) on a database D is the set of substitutions for the variables in G such that G is true with respect to $(P \cup D)$. Two queries (G, P) and (G', P') are *equivalent* if they have the same answer for all possible databases.

Two variables X and Y in a rule r are *connected* if they appear in the same predicate or if there exists in r a variable Z such that X is connected to Z and Z is connected to Y. Two predicates P_1 and P_2 appearing in a rule are *connected* if they share some variable or if there exist two connected variables appearing respectively in P_1 and P_2. A predicate p depends on a predicate q if 1) there exists a rule such that p appears in the head and q in the body or 2) if there exists a predicate s such that p depends on s and s depends on q. Two predicates p and q are mutually recursive if p depends on q and q depends on p.

A rule in a component P_i is called *exit rule* if each predicate in the body belongs to a component P_j such that $j < i$; the other rules in the component are called *recursive rules.* A recursive rule is said to be *linear* if the body of the rule contains at most one predicate mutually recursive with the head predicate. A program is linear if each rule is either an exit rule or a linear recursive rule. A linear rule is of the form $P \leftarrow L, Q, R$ where P and Q are mutually recursive predicates while L and R are conjunctions of predicates not mutually recursive with P. We will call the conjunctions L and R as *left* part and *right* part respectively.

In an *adorned program*, occurrences of predicate symbols are adorned with superscript vectors: a b (resp. a f) in the i-th position of the adornment of a predicate p denotes that the i-th argument of p is bound (resp. free). Let P be a program, and P^c be the program obtained from P by applying a rewriting method, e.g., the magic-set method or the counting method. P^c contains a new set of predicates called, respectively, *magic* and *counting* predicates. The set of rules defining the magic (resp. counting) predicates are called *magic* (resp. *counting*) *rules*, while the remaining rules are called *modified rules.*

In general, exit rules and recursive rules in an adorned program P^α have, respectively, the following form

$$p(X,Y) \leftarrow \mathrm{e}(B) \tag{1}$$
$$p(X,Y) \leftarrow \mathrm{a}(A),\ q(X_1,Y_1),\ \mathrm{b}(B) \tag{2}$$

where 1) p and q are mutually recursive predicates whose first and second arguments denote the lists of bound and free arguments, 2) $\mathbf{a}, \mathbf{b}$ and $\mathbf{e}$ are (possibly empty) conjunctions of predicates that are not mutually recursive with p and q, 3) X, Y, X_1, Y_1, A, B denote lists of variables 4) the *safety* conditions $Y \subseteq (A \cup Y_1 \cup B)$ and $X_1 \subseteq (X \cup A)$ hold. We assume also that the variables in the head are distinguished. There is no loss of generality in this assumption because each rule can be put in such a form by simple rewriting. The set of variables appearing in the right part of the rule which appear also in the left part or which are bound in the head $((X \cup A) \cap B)$ is the set of *shared variables*.

We now review the concept of query graph for an adorned program P [14, 8]. Given a query $Q = (q(a, Y), P)$ and a database D we can associate to (Q, D) a graph called *query graph* defined as follows. An arc is a triplet (a, b, c) where a and b are the source node and the destination node, while c is the label associated with the arc. Given an arc $e = (a, b, c)$ we say that the node a (resp. b) has in output (resp. input) the arc e.

Let Q be an adorned query and let D be a database, the *query graph* associated with (Q, D) is defined as follows:

1. there is an arc from x to x_1 labeled $(left, r, c)$ if there exists a ground instantiation of an adorned rule $r : p(x, y) \leftarrow \mathbf{a}(a), q(x_1, y_1), \mathbf{b}(b)$ such that $\mathbf{a}(a) \subseteq D$ and c contains the values for the shared variables in r;
2. there is an arc from y_1 to y labeled $(right, r, c)$ if there exists a ground instantiation of an adorned rule $r : p(x, y) \leftarrow \mathbf{a}(a), q(x_1, y_1), \mathbf{b}(b)$ such that $r(b) \subseteq D$ and c is the value for the shared variables in r;
3. there is an arc from x to y marked $(exit, r)$ if there exists a ground adorned rule $r : p(x, y) \leftarrow \mathbf{e}(b)$ such that $\mathbf{e}(b) \subseteq D$.

The query graph G associated with a program can be partitioned into the three subgraphs G_L, G_R and G_E containing the arcs *left*, *right* and *exit* respectively.

The *choice* construct of $\mathcal{LDL}$++ [19, 11], can be used to enforce functional constraints in rules. Thus, a goal of the form, `choice(X,Y)`, in a rule r denotes that any consequence derived from r must respect the FD $X \rightarrow Y$. In general, X can be a vector of variables — possibly an empty one denoted by " ( )" — and Y is a vector of one or more variables. As shown in [15] the formal semantics of the construct can be given in terms of negation and stable model semantics.

3 Rewriting Datalog Queries

The computation of a program rewritten by using the counting method is executed in two phases: (i) the computation of the counting set and (ii) the computation of the answer. The method can thus be viewed as stack-based, since during the first phase it remembers the number of applications of the (left part of the) recursive rule, and during the second phase it executes (the right part of) the rule an equal number of times. If there is only one recursive rule, all the elements in the list are the same and then it is sufficient to store the length of the list, according to the classical method. The presence of more than one recursive rule means that the exact sequence of rules used in the first phase must be memorized, so that the same sequence of rules, but in reversed order, can be executed during the second phase. For example, if we reached element x, starting from source node a, by the application of the left part of rules r_1, r_1, r_2, r_1, in the computation of the right part we need to apply the rules r_1, r_2, r_1, r_1.

Furthermore, if a variable in the right part of a recursive rule also appears in the left part, or it is bound in the head, then we need to know its value when computing the modified rules. This implies that we need to store in the list the values of such variables. (Recall that the pair value–rule# is also labeling the arcs of the query graph associated with the program.) The following example shows how a linear program with shared variables and more than one recursive rule is rewritten using lists.

Example 1: *Consider the following example with the query-goal* $\mathrm{p}(\mathrm{a},\mathtt{Y})$.

$$\begin{array}{lll} \mathtt{r}_0: & \mathtt{p(X,Y)} \leftarrow & \mathtt{flat(X,Z)}. \\ \mathtt{r}_1: & \mathtt{p(X,Y)} \leftarrow & \mathtt{up1(X,X_1,W)},\ \mathtt{p(X_1,Y_1)},\ \mathtt{down1(Y_1,Y,W)}. \\ \mathtt{r}_2: & \mathtt{p(X,Y)} \leftarrow & \mathtt{up2(X,X_1)},\ \mathtt{p(X_1,Y_1)},\ \mathtt{down2(Y_1,Y,X)}. \end{array}$$

The variable W in rule r_1 appears both in the left and in the right part while the variable X in rule r_3 is bound in the head and also appears in the right part. Each entry in the list defining the path contains two arguments: the identifier of the rule and a list containing the variables appearing in the right part appearing also in the left part or which are bound in the head. The resulting rewritten program (where the adornments have been omitted for brevity) is as follows:

$$\begin{array}{ll} \mathtt{c\_p(a,[\ ])} & \\ \mathtt{c\_p(X_1,[(r_1,[W])|L]))} \leftarrow & \mathtt{c\_p(X,L)},\ \mathtt{up1(X,X_1,W)}. \\ \mathtt{c\_p(X_1,[(r_2,[X])|L]))} \leftarrow & \mathtt{c\_p(X,L)},\ \mathtt{up2(X,X_1)}. \end{array}$$

$$\begin{array}{ll} \mathtt{p(Y,L)} \leftarrow & \mathtt{c\_p(X,L)},\ \mathtt{flat(X,Y)}. \\ \mathtt{p(Y,L)} \leftarrow & \mathtt{p(Y_1,[(r1,[W])|L])},\ \mathtt{down1(Y_1,Y,W)}. \\ \mathtt{p(Y,L)} \leftarrow & \mathtt{p(Y_1,[(r2,[X])|L])},\ \mathtt{down2(Y_1,Y,X)}. \end{array}$$

□

3.1 Implementation

The Pushdown method adds to each pushdown predicate an argument denoting the path connecting the initial binding to the element. Such an argument, from now on called *path argument*, is used to select the modified rule that must be used to compute the answer. For each element in the pushdown set we store the rule identifier, the list of shared variables and the address of the tuple used to compute it. In particular, we assume that each pushdown tuple is associated with an identifier, via the use of the $\mathcal{LDL}$++ choice construct. We use the notation $Id : P$ to show that "Id is the identifier for the tuple P" (Many new logic languages support the concept of *Object ID*.) Logically speaking, we can produce an equivalent program by simply making Id an additional argument of P; but the specialized notation is also suggestive of the implementation, since Id can be viewed as the address at which P is stored, and e.g., retrieving the values of attributes of P given Id is a unit cost operation.

For example, if the element b in the pushdown set is computed by using the rule r and the element a we store the tuple $(b, r, [..], Addr(a))$.

Example 2: Consider the program of Example 1. The pushdown rules are

$$\begin{array}{ll} \mathtt{c\_p(a,r_0,[\ ],nil)}. & \\ \mathtt{c\_p(X_1,r_1,[W],A)} \leftarrow & \mathtt{A:c\_p(X,\_,\_,\_)},\ \mathtt{up1(X,X_1,W)}. \\ \mathtt{c\_p(X_1,r_2,[X],A)} \leftarrow & \mathtt{A:c\_p(X,\_,\_,\_)},\ \mathtt{up2(X,X_1)}. \end{array}$$

The list associated with an element can be deduced by 'navigating' the chain defined by the last arguments. The set of rewritten rules is the following with the query-goal $\mathtt{A : c\_p(a, \_, \_, \_)}$, $\mathtt{p(Y, A)}$.

$$\begin{array}{lll}
\mathtt{r_0}: & \mathtt{p(Y,A)} \leftarrow & \mathtt{A : c\_p(X,\_,\_,\_),\ \ e(X,Y).} \\
\mathtt{r_1}: & \mathtt{p(Y,A)} \leftarrow & \mathtt{p(Y_1,B),\ \ B : c\_p(\_,r_1,[W],A),\ \ down1(Y_1,Y,W).} \\
\mathtt{r_2}: & \mathtt{p(Y,A)} \leftarrow & \mathtt{p(Y_1,B),\ \ B : c\_p(\_,r_2,[X],A),\ \ down2(Y_1,Y,X).}
\end{array}$$

□

Observe that here when we compute the predicate $B : c\_p(...)$ the variable B is bound and this corresponds to a direct access to the memory. Thus, the method is very similar to the *Bushy-Depth-First* method used in the implementation of $\mathcal{LDL}$ [19]. Notice also that the shared variables which appear also in the head could be omitted from the list of share variables since they appear also in the predicate pushdown appearing in the body of modified rules.

3.2 Cyclic Databases

The counting method is unsafe if the left part of the query graph is cyclic. Various techniques have been proposed to deal with these situations, including the *magic-counting* [14], that combines the counting method and the magic-set method, and more specialized algorithms [6, 8, 1]. Here, we will use the FD defined by choice goals in the rules, to avoid the generation of an infinite number of tuple identifiers. For instance, the rewritten program corresponding to the program of Example 3, is as follows:

$$\begin{array}{ll}
\mathtt{B : c\_p(a,r_0,[\ ],nil).} \leftarrow & \mathtt{fd(a,B).} \\
\mathtt{B : c\_p(X_1,r_1,[W],A)} \leftarrow & \mathtt{A : c\_p(X,\_,\_,\_),\ \ up1(X,X_1,W),\ \ fd(X_1,B).} \\
\mathtt{B : c\_p(X_1,r_2,[X],A)} \leftarrow & \mathtt{A : c\_p(X,\_,\_,\_),\ \ up2(X,X_1),\ \ fd(X_1,B).}
\end{array}$$

$$\begin{array}{lll}
\mathtt{r_0}: & \mathtt{p(Y,A)} \leftarrow & \mathtt{A : c\_p(X,\_,\_,\_),\ \ e(X,Y).} \\
\mathtt{r_1}: & \mathtt{p(Y,A)} \leftarrow & \mathtt{p(Y_1,B),\ \ B : c\_p(\_,r_1,[W],A),\ \ down1(Y_1,Y,W).} \\
\mathtt{r_2}: & \mathtt{p(Y,A)} \leftarrow & \mathtt{p(Y_1,B),\ \ B : c\_p(\_,r_2,[X],A),\ \ down2(Y_1,Y,X).}
\end{array}$$

where the rule defining fd is as follows:

$$fd(X,Y) \leftarrow choice((X),(Y))$$

The fd goals in the rules avoid the firing of rule instances that differ only in the values of the tuple identifiers.

3.3 The Linear Pushdown Algorithm

We next present the pushdown algorithm for linear programs.

Algorithm 1 *[Linear Pushdown Rewriting]*
Input: *Query* $(q(a,Y),P)$ as in Algorithm 1.
Input: *Adorned query* $(q(a,Y),P)$ where the rules have form

$$\begin{array}{l}
p(X,Y) \leftarrow \mathbf{e}(B) \\
p(X,Y) \leftarrow \mathbf{a}(A), q(X_1,Y_1), \mathbf{b}(B)
\end{array}$$

Output: *Rewritten query* $((A : c\_q(a,\_), q(Y,A)), P^{ec})$
Notation: $C_r = A \cap B$.

```
begin
    $P^{ec} := \{ \}$
    % Generate Pushdown Rules
    $I : P^{ec} := P^{ec} \cup \{ I : c\_p(a, (r_0, [\,], nil)) \} \leftarrow fd(a, I) \}$
    for each recursive rule $r$ s.t. $X \neq X_1$ do
      if $Y = Y_1$ and $p = q$ then
        $P^{ec} := P^{ec} \cup \{ I : c\_p(X_1, (r, C_r, J)) \leftarrow c\_p(X, (r, C_r, J)),\ \mathbf{a}(A),\ fd(X_1, I) \}$
      else
        $P^{ec} := P^{ec} \cup \{ I : c\_p(X_1, (r, C_r, J)) \leftarrow J : c\_p(X, \_),\ \mathbf{a}(A),\ fd(X_1, I). \}$
    % Generate Modified Rules
    for each exit rule do
      $P^{ec} := P^{ec} \cup \{ p(Y, I) \leftarrow I : c\_p(X, \_), \mathbf{e}(X, Y), \}$
    for each recursive rule $r$ s.t. $Y \neq Y_1$ do
      if $X = X_1$ and $p = q$ then
        $P^{ec} := P^{ec} \cup \{p(Y, I) \leftarrow p(Y_1, I), I : c\_p(X, (r, C_r, J)),\ \mathbf{b}(B)\}$
      else
        $P^{ec} := P^{ec} \cup \{p(Y, I) \leftarrow p(Y_1, J), J : c\_p(\_, (r, C_r, I)),\ \mathbf{b}(B)\}$
end.
```

The predicate $I : c\_p(X, \_)$ in the body of the modified recursive rules can be omitted if no bound variable in the head appear in the right part of the body.

Theorem 1: *Let $Q = (G, P)$ be an adorned query. Let Q' be the query obtained by application of algorithm 1 to Q. The computation of the fixpoint of the rewritten program Q' always terminates.* □

Theorem 2: *Let $Q = (G, P)$ be an adorned query. Let Q' be the query obtained by application of algorithm 1 to Q. Then Q and Q' are equivalent.* □

4 Complexity and Experimental Results

Let $Q = \langle G, P\rangle$ be a query and let D be a database. The query graph $G = \langle V, E\rangle$ associated with (Q, D) is composed of the three subgraphs $G_L = \langle V_L, E_L\rangle$, $G_E = \langle V_E, E_E\rangle$ and $G_R = \langle V_R, E_R\rangle$ such that $V_L \cup V_R = V$ and $E_L \cup E_E \cup E_R = E$. Let $m_i(L)$ and $m_i(R)$ be the number of arcs with labels $(left, r_i, [..])$ and $(right, r_i, [..])$ respectively. Let $n(L)$ and $n(R)$ be the global number of nodes in G_L and G_R and let $m(L)$ and $m(R)$ be the global number of arcs in G_L and G_R, then

$$m(L) = \textstyle\sum_{i=1}^{r} m_i(L) \qquad\qquad m(R) = \textstyle\sum_{i=1}^{r} m_i(R)$$

where r is the number of recursive rules in P.
We assume that the costs of the operations are the following:

1. The cost to access a tuple $p(x, x1)$ is equal to the value of the cost access function $h(m)$ which depends on the access method and on the number of p-tuples m. In particular, the function, $h(m)$ is equal to (i) 1 when the address (tuple-identifier) of the tuple is known, (ii) a constant $c > 1$ when the key of the tuple is known, and (iii) $O(m)$ when the tuples are accessed sequentially. To distinguish access to base and derived relations we shall denote the cost access functions h_B and h_D respectively.

2. Given two sets S_1 and S_2, the costs for union and difference are linear in the size of the sets involved and are both equal to $h(m_1) \times m_2$ where $m_1 = min(|S_1|, |S_2|)$ and $m_2 = max(|S_1|, |S_2|)$. Moreover, if there are no indexes on the sets then we use sequential access and the cost for both union and difference is $O(m_1^2 \times m_2)$.

To compute the complexity we use the same-generation program

```
sg(X,Y) ← flat(X,Y).
sg(X,Y) ← up(X,X1), sg(X1,Y1), down(Y1,Y).
```

We assume that the number of tuples and constant in the base relations are comparable, i.e. $O(m(L))=O(m(R))=O(m)$ and $O(n(L))=O(n(R))=O(n)$.

Semi-naive Fixpoint: Since the recursive predicates have $O(n^2)$ tuples, the cost of adding a tuple is $O(h_D(n^2))$, while the cost of accessing a tuple from a base relation is $h_B(m)$. The computation corresponds to navigating both the left and the right parts of the query graph: thus, for each arc in the right part of the graph (accessed at cost $h_B(m)$) $O(m)$ arcs are selected from the left part, where the cost for each arc is also $h_B(m)$. Therefore the global cost is

$$O(m^2 \times (h_B^2(m) + h_D(n^2)).$$

Yannakakis' Method: The method proposed in [18] improves the computation of the fixpoint for chain programs, i.e., programs with binary predicates not containing shared variables. This method has complexity

$$O(m \times n \times h_D(n^2)).$$

In the present form, the method has very limited practical interest due to the fact that it does not make use of the bound query arguments (same as for the seminaive computation) and it is based on the construction of a ground program graphs of exceedingly large sizes.

We will next concentrate on methods that make effective use of the query constants. Therefore, let $\hat{m}$ and $\hat{n}$ respectively denote the number of tuples and constant in the database used to compute a bound query. That is, given a query $Q = <q(a, Y), P>$ and a database D, then $\hat{m} = \sigma(Q, D)$ and $\hat{n} = \sigma(Q, H)$ where σ is a *selection function* which takes in account the fact that the rewriting of the query allows us to use a restricted portion of the database ($0 < \sigma(Q, D) \leq m$ and $0 < \sigma(Q, H) \leq n$). Thus, although $O(\hat{m}) = O(m)$ and $O(\hat{n}) = O(n)$ we will use $\hat{m}$ and $\hat{n}$ when a restricted portion of the database is used.

The Magic-set method: The rewritten program consists of two sets of rules, called *magic* and *modified* rules respectively:

```
m_sg(a).                            sg(X,Y) ← m_sg(X), flat(X,Y).
m_sg(X1) ← m_sg(X), up(X,X1).      sg(X,Y) ← m_sg(X), up(X,X1),
                                              sg(X1,Y1), down(Y1,Y).
```

The number of tuples of the magic predicates (*m_sg* in our example) are bound by $O(\hat{n})$ while the number of tuples of the modified predicates (*sg* in our example) are bound by $O(\hat{n}^2)$. The computation of the magic rules takes cost $O(\hat{m} \times (h_B(m)+h_D(\hat{m}))$ since the number of body-tuples (satisfyed body conjunctions) is bound by $O(\hat{m})$ and each head-tuple must be added to a 'magic relation' with cost $O(h_D(\hat{m}))$. The factor $h_B(m)$ is the cost to access the tuples of the base relations (*up* in our example).

For the computation of the modified rules the only difference with respect to the semi-naive strategy is that we have an additional predicate in the body of the rules. The cost to produce a new tuple is now $h_B^2(m) \times h_D(\hat{m}) \times h_B^2(m)$, where $h_D(\hat{m})$ and $h_B(m)$ are the costs to access the tuples of the 'magic' and base relations. Therefore, the global cost is:

$$O(\hat{m}^2 \times (h_B^2(m) \times h_D(\hat{m}) + h_D(\hat{n}^2)).$$

The Supplementary Magic-set method: The method is similar to the magic-set one; however a supplementary relation is used in the modified rules, as follows:

```
sg(X,Y) ←        s_sg(X,X1),  sg(X1,Y1),  down(Y1,Y).
s_sg(X,X1) ←   m_sg(X),  up(X,X1).
```

The size of the supplementary relation (*s_sg* in our example) is bound by $O(\hat{m})$ since it contains a subset of the tuples in the database (*s_sg* contains a subset of the tuples of *up*). The computation of the set of magic and supplementary rules has cost $O(\hat{m} \times (h_B(m) + h_D(\hat{m}))$.

For the computation of the modified rules we need to access the base relations of the right part of the rules (the relation *down* in our example) (at cost $h_B(m)$) and the supplementary relation (at cost $h_D(\hat{m})$). The global cost is then

$$O(\hat{m}^2 \times (h_B(m) \times h_D(\hat{m}) + h_D(\hat{n}^2))$$

The Acyclic Counting Method: The rewritten program consists of two sets of rules, called *counting* and *modified* rules, respectively, which for the example at hand define predicates *c_sg* and *sg* as follows:

```
c_sg(a,0).
c_sg(X1,I+1) ←   c_sg(X,I),  up(X,X1).
sg(Y,I) ←          c_sg(X,I),flat(X,Y).
sg(Y,I−1) ←    sg(Y1,I),  I > 0,  down(Y,Y1).
```

The sizes of the counting and modified relations are both bound by $O(\hat{n}^2)$. The computation of the counting rules terminates in $O(\hat{n})$ steps and each iteration has cost $O(\hat{m} \times h_B(m))$ since $O(\hat{m})$ is the number of tuples used in the database and $h_B(m)$ is the cost to access a single tuple. The cost to add the new tuples is constant since these have an incremented index and we do not need to check for membership. The computation of the counting rules has then cost $O(\hat{n} \times \hat{m} \times h_B(m))$ and, thus, the computation of the modified rules has also cost $O(\hat{n} \times \hat{m} \times h_B(m))$. The global cost is then

$$O(\hat{n} \times \hat{m} \times h_B(m))$$

In passing, we also mention the extension proposed by [8] with complexity $O(n^2 \times m)$, that of [6] with complexity $O(n \times m)$, and that of [1] which has complexity $O(n^3)$. None of these proposals is based on some simple modification of the original algorithm; rather they use completely new algorithms.

	Range of applic.	Complexity
Semi-naive	General	$O(m^2 \times (h_B^2(m) + h_D(n^2))$
Yannakakis	Restricted Linear	$O(n \times m \times h_D(n^2))$
Magic-set	General	$O(\hat{m}^2 \times (h_B^2(m) \times h_D(\hat{m}) + h_D(\hat{n}^2))$
Suppl. magic	General	$O(\hat{m}^2 \times (h_B(m) \times h_D(\hat{m}) + h_D(\hat{n}^2))$
Counting	Restricted Linear	$O(\hat{m} \times \hat{n} \times h_B(m))$
Pushdown	All Linear	$O(\hat{m}^2 \times (h_B(m) + h_D(\hat{n}^2))$

Figure 1: *Asymptotic Complexity*

	m = 1000	m = 2000	m = 3000	m = 4000	m = 5000
Magic-set	407.33	547.39	671.30	791.14	891.66
Suppl. magic	21.64	33.29	43.39	53.06	61.29
Pushdown	7.03	13.40	18.90	24.06	28.34

Figure 2: *Experimental Results - sequential access* - $n = 200, \hat{n} = 50, \hat{m} = 100$

The Linear Pushdown Method: The rewritten program consists of two sets of recursive rules, called *pushing* and *modified* respectively, which for the example at hand define predicates *c_sg* and *sg* as follows:

$$\mathtt{B : c\_sg(a, nil).} \leftarrow \quad \mathtt{fd(a, B)}$$
$$\mathtt{B : c\_sg(X_1, A)} \leftarrow \quad \mathtt{A : c\_sg(X, \_),\ \ up(X, X_1),\ \ fd(X_1, B).}$$

$$\mathtt{sg(Y, A)} \leftarrow \quad \mathtt{A : c\_sg(X, \_),\ \ e(X, Y).}$$
$$\mathtt{sg(Y, A)} \leftarrow \quad \mathtt{sg(Y_1, B),\ \ B : c\_sg(\_, A),\ \ down(Y_1, Y).}$$

The number of tuples in the pushing relations is bound by $O(\hat{m})$, and the number of tuples in the modified relations is bound by $O(\hat{n}^2)$. The computation of the pushdown rules takes cost $O(\hat{m} \times h_B(m))$ because it is equivalent to the navigation of a graph with $\hat{m}$ arcs.

For the computation of the modified rules we need to access the base relations appearing in the right part of the rules (the relation *down* in our example) at cost $h_B(m)$ and the pushing relations (*c_sg* in our example) at cost 1. The cost to store a tuple is equal to $h_D(\hat{n}^2)$. The global cost is then

$$O(\hat{m}^2 \times (h_B(m) + h_D(\hat{n}^2))$$

The complexity results are resumed in the Table of Figure 1.

5 Conclusion

Due to space limitations, we can only give a short summary of the results of experiments discussed in [5]. These experiments confirm that the supplementary magic method is an order of magnitude better than the magic set method when sequential access is used. When extensible hashing is used for storing and retrieving tuples, then the latter only brings a modest 20% improvement with respect to the former. The pushdown method, however, is consistently four time faster than the supplementary magic method. The tables of Figure 2 and Figure 3 give typical results obtained on a PC with a 25-MH Intel 486 CPU.

	m = 1000	m = 2000	m = 3000	m = 4000	m = 5000
Magic-set	24.55	25.10	25.54	26.09	26.59
Suppl. magic	21.37	21.81	22.14	22.46	22.79
Pushdown	5.05	5.22	5.38	5.55	5.66

Figure 3: *Experimental Results - Extensible Hash* - $n = 200, \hat{n} = 50, \hat{m} = 200$

References

[1] H. Aly and Z.M. Ozsoyoglu. Synchronized counting method. In *Proc. of the Fifth Int. Conf. on Data engineering*, pp 366-373, 1989.

[2] F. Bancilhon, D. Mayer, Y. Sagiv, J.F. Ullman. Magic sets and other strange ways to implement logic programs. In *Proc. Fifth ACM PODS*, pp 1-15, 1986.

[3] F. Bancilhon and R. Ramakrisnhan. Performance evaluation of data intensive logic programs. *Found. of Ded. Dat. and L. Progr.*, pp 439-518, 1988.

[4] C. Beeri and R. Ramakrisnhan. On the power of magic. *Journal of Logic Programming*, 10 (3 & 4), pp 255-299, 1991.

[5] S. Greco and C. Zaniolo. Efficient Execution of Recursive Queries Through Controlled Binding Propagation. *Technical Report DEIS*, 1993.

[6] R. Haddad and J. Naughton, A counting algorithm for a cyclic binary query. *Journal of Computer and System Science*, 43(1):145-169, 1991.

[7] B. Lang. Datalog Automata. In *Proc. of the 3rd Int. Conf. on Data and Knowledge Bases* Jerusalem, Israel, March, 1988, pp 389–401.

[8] A. Marchetti-Spaccamela, A. Pelaggi, D. Saccà. Comparison of methods for logic query implementation. *J. of Logic Programm.*, 10, pp 333-361, 1991.

[9] J. Naughton, R. Ramakrisnhan, Y. Sagiv, J.F. Ullman. Argument reduction by factoring. In *Proc. of the 15th VLDB Conference*, pp 173-182, 1989.

[10] J. Naughton, R. Ramakrisnhan, Y. Sagiv, J.F. Ullman. Efficient evaluation of right-, left-, and multi-linear rules. In *Proc. of the SIGMOD Conf.*, 1989.

[11] R. Ramakrisnhan, D. Srivastava, S. Sudanshan. CORAL — Control, Relations and Logic. In *Proc. of 18th VLDB Conference*, 1992.

[12] R. Ramakrisnhan, Y. Sagiv, J. Ullman, M. Vardi. Logical Query Optimization by Proof-Tree Transformation. In *JCSS.*, No. 47, pp 222-248, 1993.

[13] D. Saccà and C. Zaniolo, The generalized counting method of recursive logic queries for databases. *Theor. Computer Science*, No. 62, 1988, pp 187-220.

[14] D. Saccà and C. Zaniolo, Magic-counting methods. In *Proc. of the 1987 ACM SIGMOD Int. Conf. on Management of Data*, pp 149-59, 1987.

[15] D. Saccà and C. Zaniolo. Stable models and non-determinism in logic programs with negation. *Proc. of the Ninth ACM PODS*, pp 205-217, 1990.

[16] J.F. Ullmann. *Principles of Data and Knowledge-Base Systems.* Volule 1 & 2, Computer Science Press, New York, 1988.

[17] L. Vielle. Recursive Query processing: The Power of Logic. *Theoretical Computer Science*, 1989.

[18] M. Yannakakis. Graph-Theoretic Methods in Database Theory. In *Proc. of the Ninth ACM PODS*, 1990, pp 230–242.

[19] C. Zaniolo. Design and implementation of a logic based language for data intensive applications. In *Proc. Int. Conf. on Logic Programming*, 1988.

Towards a Dynamic Multi-Agent Organization

Emmanuelle Le Strugeon, René Mandiau, Gaëtan Libert

LAMIH, University of Valenciennes, Le Mont Houy,
BP 311, 59304 Valenciennes Cedex, France.

Abstract. A method enabling the dynamic organization of multiagent systems is proposed. It is based on studies about performance of different organizations when they have to handle various situations. The method uses a library of organizational models and three selection criteria relative to the current task the multiagent has to achieve. The agents apply the selected organizational model by the propagation of behavioral rules among them. Endly, some implementation aspects are presented.

1 Introduction

The organization of multiagent systems is one of the more discussed problematics in the Distributed Artificial Intelligence (DAI) field. The discussion comes from the confrontation between two opposed approaches, depending on the way organizations are created. In the first approach, organizations are designed at the beginning to handle one specific kind of problem. The second approach favours the flexibility of the organization, to be able to face different situations. The purpose of this work is an attempt to issue a third approach, integrating the advantages of the two preceding approaches.

In the first part of this paper, the notion of organization is explained and the respective advantages of the existing approaches are presented. The second part introduces a new way of conceiving multiagent organizations, requiring the study and the comparison of different organizations. In the third part, details are given about a method realizing this new approach. In the fourth part, improvements and implementation aspects are discussed.

1.1 Notion of Organization

Distributed AI is based on the notion of collective intelligence. The intelligence is searched, not in increasing the reasoning complexity of one entity (as in “classical” AI), but in improving the interaction modes among several entities, called agents, in the aim to have a global statisfying result.

A generic definition for these entities has been given recently (see, for example, [7]): agents are AI systems or humans. A MultiAgent System (MAS) is composed of agents. The multiagent system is immersed in an environment, composed of all objects that are not agents and that agents can describe and act on. A MAS and its environment form a multiagent world. However, these elements can be classified and designed according to three different abstraction levels [13]:

- At the lowest level, the internal structure of one agent is detailed.

- At the intermediary level, multiagent systems are represented as sets of agents, interacting, together and with objects of the environnement.
- At the highest level, the interactions between a MAS and its environment can be studied.

The work presented in this paper focus on agents that cooperate to achieve a common global task. Because of this common task, the agents have to coordinate their activity: they have to attribute actions to agents able to perform them, and to coordinate the execution of these actions. In these aims, they have to communicate to come to an agreement. The organization is the structure supporting coordination, task attribution and communication among agents. Organization thus belongs to the second intermediary level of MAS design.

1.2 Static or Dynamic Organizations

Two differents approaches exist in the design of MAS organization, depending on the way it is conceived of, either static or dynamic.

In the static approach, the organization is fixed a priori, during the system's design process. The rules that structure exchanges are the same whatever the situation is. Such organizations are generally composed of cognitive agents, that possess models of themselves and of the others. As a consequence, their exchanges are rather complex and of a hig abstraction level; the result of their interactions is foreseeable. However, the organization is designed in the aim to solve one specific kind of problems, it can hardly address new tasks [15].

The dynamic approach, based on reactive agents, shows MAS that are not initially organized. The organization has a kind of dynamism because of its “emergence”: the structure is created in response to perturbations coming from the environment. Among these agents, the organization does not result from a common will but from the sum of their individual behaviours [18,4]. Such organizations are more flexible than the former, because each agent adapts its behaviour to the environment. However, the resulting organizational structure is difficult to foresee: programming specific individual rules does not necessarily leads to the desired system's behaviour.

The advantages of the both approaches can be combined. The dynamism of the organization can be achieved differently, enabling the agents to take the organization which is the best appropriate in their current situation. The organization can be seen as a process including several states. The dynamism consists in the reorganization of the system, that means the transition from a state to another. The problem lies in the fact that this reorganization have to be done by the agents themselves. So, they have to possess the means to realize this dynamic organization.

Three means have been listed:

- The agents have access to a data base containing all the organization modes they can take. This data base is called library of organizational models.
- They have a method to select one of the library's elements.
- They have a method to get organized according to the model selected in the library.

2. Study of Organizations' Characteristics

In order to create the library, classifications of organizations have been studied. The method to select organizational models is based on studies about the performance of different organizations in various contexts.

2.1 Attempts to Classify Organizations

The knowledge about organizations comes from the study of their similarities and differences, and thus from their classification. Because a taxonomy of the organizational structures does not exist yet, we decided to present two DAI classification methods: Fox's and Gasser's classifications.

Fox [81] classifies organizations along a decision making line, from the simplest one, realized by a single agent. The group made of independent and non cooperating agents is only a little more complicated. Then, the following organizations come, in this order: the simple hierarchy (one chief gives orders to a set of executing agents), the uniform hierarchy (several levels of decision making), the multidivisional hierarchy (decisions are spread among sub divisions), a price system (negociation among separate organizations), collective organization (negociation with long term contracts), general market (individual goals).

Gasser [GAS 92] classifies DAI organizational structures in four categories: centralized (or hierarchical), market-like, pluralistic community and community with rules of behaviour.

A hierarchical organization is a pyramid-shape configuration, including master-slave links between agents of different hierarchical levels. At each level, "masters" centralize decision-making and control powers.

Pluralistic communities are composed of independent agents who prepare solutions to the problems and then communicate their results to the other members of the community who have to assess or to improve them [9,11].

Contract's principle is the basis of market-like organizations: an agent (the "manager") broadcasts an invitation to bid to the group or only to a part of the group. Some of the agents offer their services. The manager agent selects one of them and awards a contract. Such organizations consist in independent agents who solve their individual tasks by submitting sub-tasks to the others (the reference example is the Contract-NET [17].

The last type of organization, the community with rules of behaviour is a community of specialists who interact according to defined protocols. Multi-expert systems and multi-knowledge sources systems belong to this category.

2.2 Performances of Organizations

To be able to choose an appropriate organization in the current situation, knowledge about the different organizations' features is necessary. Comparisons and evaluation of different real or multi-agent organizations exist in DAI and social sciences literature.

The performance of an organization is generally expressed in terms of production costs or efficiency, by sociologists and economists. It is estimated using criteria that generally include the task's achievement duration and the number of communications exchanged. With the obtained performances, conclusions have been drawn about the adequation of the organizations, relatively to the context. Seven different studies are collected in the table 1.

Table 1. Organizations evaluations.

authors	studied organizations	comparison criteria	type of the appropriate organization according to criteria
Malone [14]	hierarchy(s), market(s)	production, coordination, vulnerability	production -> market coordination -> hierarchy
Cohen [3]	hierarchy and market	coordination	coordination -> hierarchy
Numaoka [16]	hierarchy and market	global / individual goal	large scale task -> hierarchy
Fox [5]	5 organisations	complexity, uncertainty	complexity -> heterarchy uncertainty-> hierarchy
Kumar [10]	4 organizations with a progressive decentralization	performance	*depending on:* role/objective compatibility, information availability, consistency of solutions
Bouron [1]	organizations with agents of different social complexity	efficiency (performance and cost)	*depending on:* activity, communication selectivity, agents distribution
Chevrier [2]	free organization, hierarchy and market	solving duration, number of communications	communication -> market

To make a synthesis from these studies seems at first impossible, because the organizations are evaluated according to criteria that are different. However, some notions arise from the whole set. Especially, they show that:

- The perfect organization that would be the best in every situation does not seem to exist. Ishida [8] asserts that there is no single organization able to handle correctly every problem and every environmental conditions. Various situations require different organizational solutions.
- In a given situation, there is generally more than one appropriate organization. This second point is a corollary of the first one.
- Several possible solutions to select the appropriate organization, according to the context, are provided. Three main criteria arise from these studies about coordination, action and knowledge.

  The *coordination* criterion is the most obvious of them. It seems that more a task requires the coordination of a high number of constrained actions, more the organization needs to be hierachically structured (Malone, Cohen).

  The *action* criterion represents the scale of the task, the number of actions needed to achieve it (Numaoka).

The *knowledge* criterion takes into account what have to be known to perform the task. Information availability (Kumar), uncertainty (Fox) and vulnerability (Malone) criteria are grouped in this last criteria.

3 Proposition of A Method to Realize the Dynamic Organization

Means that agents need to behave according to a dynamic organization listed in paragraph 1.2, are proposed in the next part.

3.1 Creation of an Organizational Models Library

The library's elements are organizational models. A model prescribes to the agents how they must behave in the group. It describes the roles that components play in the global structure and the relations among them. It specifies, for each member of the group, the components with which it can communicate, the type of authority relation and the cooperation mode to adopt toward each of them. Organizations are seen as structures of components among which some authority (hierarchical) and communication relations exist [12].

The models chosen to form the library are the four organization models of Gasser's classification, described previously (§2.1): the hierarchy, market, community and society (community with rules of behaviour) structures. This choice is going to be improved: models will be added or removed if it is necessary.

3.2 Selection of a Model

The selection of a structure for the group is realized by evaluating, among different given structure models, the one that best suits the current situation. It is based on the characteristics of the global task to perform. The characteristics used for the selection correspond to the three criteria distinguished while evaluating organizations' performance. The data used to evaluate these characteristics are provided by the realization plan of the task.

Notations used in the following formulae are:

- The set of all tasks in the plan is *Tasks*. Any element of this set is noted T_j. *Precond*(T_j) is the set of conditions that must be satisfied to achieve the task T_j. Conditions are couples *(object, state)*.
- Among the tasks of the plan, some are primitive tasks, achievable by one agent, the set of such tasks is noted *Prim*.
- obj_k is an object k of the environment. *state_of*(obj_k) is the state of obj_k.
- *card* is the cardinal number function (number of elements in a set). *card(Tasks)* is shorten in *T*, *card(Prim)* in *P*.

The knowledge characteristic (K) represents the amount of facts that must be known for the task to be achieved. An estimation of it is the ratio of the different objects which state must be known out of the number of tasks in the plan *(1)*.

$$\mathrm{K} = \frac{\mathrm{card}\left(\left\{\mathrm{obj}_k / \exists \mathrm{T}_j \in \mathrm{Prim}, \left(\mathrm{obj}_k, \mathrm{state\_of}\left(\mathrm{obj}_{jk}\right)\right) \in \mathrm{precond}\left(\mathrm{T}_j\right)\right\}\right)}{\mathrm{T}} \qquad (1)$$

The action characteristic (A) gives a measure of the number of primitive tasks in the plan in relation to the total number of tasks *(2)*.

$$\mathrm{A} = \frac{\mathrm{P}}{\mathrm{T}} \quad (2)$$

The coordination characteristic (R) results from the evaluation of the coordination "complexity" of the plan, i.e. the amount of constrained tasks. If the plan is represented as a graph, in which each node is a task, the graph density —measure of the number of nodes in the graph representation— is taken as an estimation of this complexity *(3)*.

$$\mathbf{R} = 1 + \frac{1 - \mathrm{P}}{(\mathrm{T} - 1)\left(\frac{1}{2}\mathrm{T} - 1\right)} \qquad (3)$$

These formulae have been used to classify ten graph representations of tasks. This result will be later published and explained.

The measures of task's demands are normed and represented in a space. The localization of any task on this diagram should enable to classify it as one of the generic tasks, represented by the cube apexes (fig. 1).

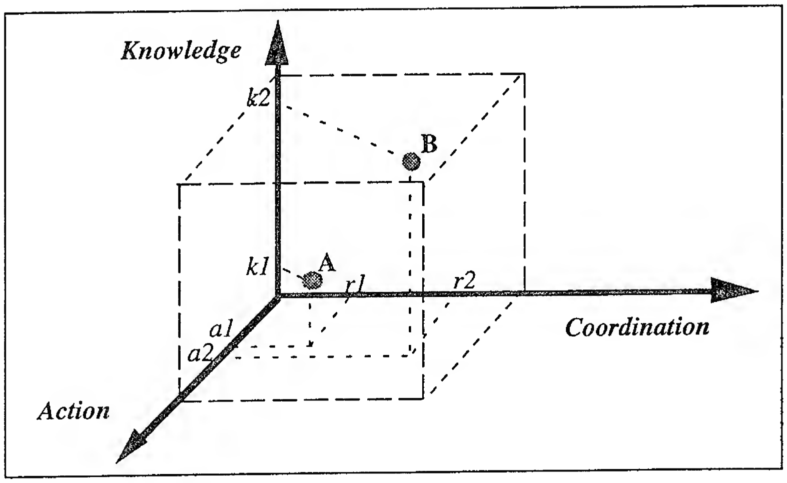


Fig. 1. Comparison between two tasks's demands.

Each apex, seen as a generic task, is associated to a component of the model's library. These associations are founded on the studies of organizations performances presented in part 2.2. From the four models' characteristics, the characteristics of the tasks they are appropriate for, are determined.

The *hierarchy* (H) supports properly situations demanding a high coordination. That means a high R characteristic of the task. Constraints imposed by authority links lead to grouping operational efforts in a little number of components. Hierarchy is good only for low A situations.

The *community* (C) allows an important distribution of tasks among agents, and thus the efficient execution of numerous actions (high A). Coordination is more difficult to achieve than in a hierarchy, so the community suits only to low R cases. For this type of organization, we have no result allowing us to rule on its aptitudes about knowledge. Indeed, two opposite aspects exist. On the one hand, communications seem to be less efficient in community than in other organizations, because of the lack of structure (exchanges are less "targeted"). On the other hand, knowledge diffusion is considered to be better in it, because of the communicational net density. In this uncertainty about the knowledge characteristic, this type of organization has been reserved for low K situations only.

The *market* (M) favours communication selectivity and enables its use for high K situations. It is advised when a lot of actions have to be performed (high A) and when coordination is poor (low R).

The *society* (S) have characteristics which are similar to community's, with a better aptitude for coordination and knowledge diffusion, thanks to the use of specific rules sparing numerous preliminary communications. Consequently, it is correct for high demands situations (high K, A and R).

The cube representation that was previously used, is usable again to display these conclusions. Letters symbolizing organizational models are situated at the key points of the cube, the apexes (fig. 2). For the point (0,0,0), it is useless to choose an organization. Indeed, when the task has little demands, the current organization is supposed to be able to handle this new situation whithout any change.

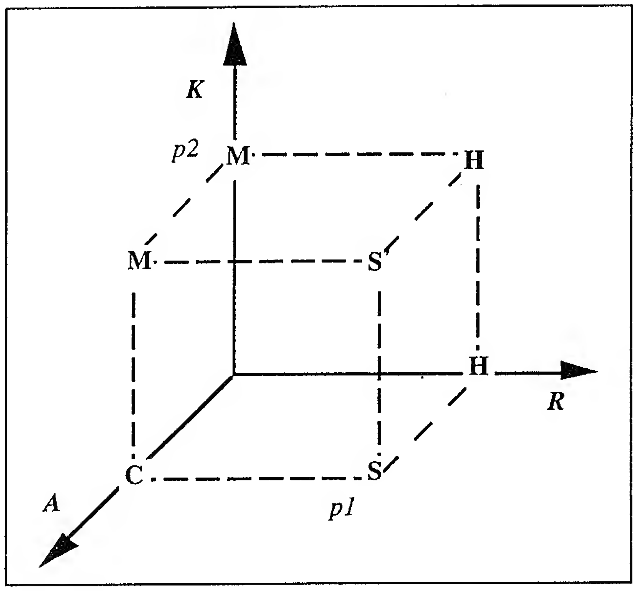


Fig. 2. Organization models/Generic tasks associations.

A filling process of the cube representation, that follows organizations' aptitudes, let two points aside: the point p1 (low K, high A, high R) and the point p2 (high K, low A and R). We have arbitrarily chosen to put the "society" model at the point p1 and the "market" model at p2, because they have the qualities required by their tasks achievement.

One must notice that this describes a method and not a solution. The library presented includes only four models, but others may be added. The consequence of models addition or withdrawal would significantly modify the associations between generic tasks and organizational models, that should be evaluated again.

3.3 Diffusion of an Organizational Behaviour Among Agents

When one organizational model is selected, the agents structures their group. A reorganization is possible if the agents can create new links and break old ones. The organization model provides a way to build the new MAS's frame. It gives behavioral instructions, that may be seen as "roles" from an individual agent's point of view. Roles define which are the specific authority and communication relations that agents are allowed to build and use. Three methods have been envisaged.

The "all or nothing" method consists in breaking all existing links and in setting all agents at the same level: the whole system is reset. The new created organization corresponds perfectly to the chosen model, but the agents can not structure themselves if all communication links are cut.

The "ecological" method consists in basing the new organization on the old one. A maximal number of links and components are kept. The new structure develops from fixed elements. The advantage lies in the saving in construction and destruction actions. The disadvantage is the need of a global knowledge about the real current organization. It is more difficult to determine which elements will be kept, than to break the whole system and then to build the new one as in the previous method.

The iterative method combines both previous methods. Its principle is the same as the "ecological" method's, except that the invariant elements are not searched in a global manner. The organization is spread with the help of the actual net, which is modified as agents acquire new roles. Its saving looks like the "ecological" method's, but an agent can be left alone because all its accointances cut all the links that they had with it. This problem is easily solved in implementing a behavioral rule preventing the destruction of a link without the other agent's authorization.

This latter method affords, in our opinion, the best compromise.

3.4 Early Implementation Steps

An experimental multi-agent system is currently developed. Another paper explains the representations and data strucures used for the implementation [13]. However, the broad lines of the system's implementation can be expressed.

We aim more at realizing a useful testbed for the laboratory research works upon MAS, than a dedicated system for organizational experiments. The implementation needs included: simulation of agents with a message passing technique; high facilities in realizing changes in agents' structures and numbers, environment definition and library's contents.

Message passing technique is easy to simulate in the Ada language. It is, moreover, a readable and easy to maintain language. Ada_was chosen for these reasons. Agents are simulated by Ada tasks. They act on an environment that is also an Ada task. Messages are exchanged through mail boxes (communications are asynchronous). Organization propagation is concretized by the diffusion of behavioral indicating "algorithms" among agents.

To simplify maintenance, the whole system is made of many separate units: MAS, environment, planner, organization tools and communication tools are distinct. Interfaces help in implementing eventual changes.

Drawbacks of these technical choices is that our system is not a real time system, since the MAS does not "react" instantaneously to environmental events. The MAS's reaction depends on the agents' perception activity. Agents perceive regularly their environment to update their model of it, that is why environmental changes have no effect during the period comprised between two perception acts.

4 Conclusion

This paper proposes a method to realize the dynamism of multiagent systems organizations.

Fundamental characteristics of the organizations are studied. On this basis, three means are distinguished to enable agents to organize themselves. The first tool consists in a library composed of organizational models. The second one is a method to select one of the library's models according to the current task characteristics. The third tool is a method for the agents to create a new organization, on the base of the selected model.

The realization of the global organization method is explained, but it needs further improvements and evaluations, with the help of the system which is currently developped. Moreover, a formalism to specify the problems we address is studied.

References

1. T. Bouron: Structures de communication et d'organisation pour la coopération dans un univers multi-agents. Ph. D. Thesis, Paris VI, 17 Nov. 1993.
2. V. Chevrier: Etude et mise en oeuvre du paradigme multi-agents : de Atome à Gtmas. Ph. D. Thesis, Nancy I, INRIA Lorraine, 11 Juin 1993.
3. M.D. Cohen: Artificial Intelligence and the dynamic performance of organizational designs. In: J. March, R. Weissi Baylon (eds.): Ambiguity and command. Marshfield, USA: Pitman, 1986.
4. J. Ferber, A. Drogoul: Using reactive multi-agent systems in simulation and problem-solving. In: N. Avouris , L. Gasser (eds.): Distributed Artificial Intelligence: Theory and Praxis. The Netherlands: Kluwer Academic Publishers, 1992.
5. M.S. Fox: An organizational view of distributed systems. IEEE Transactions on SMC 11, n°1, 70-79 (1981).

6. L. Gasser: An overview of Distributed Artificial Intelligence. In: N.M. Avouris & L. Gasser (eds.): Theory and Praxis. The Netherlands: Kluwer Academic Publishers, 1992.
7. T.J. Grant: A review of Multi-Agent Systems techniques, with application to Colombus User Support Organisation. Future Generation Computer Systems 7, 413-437 (1991/1992).
8. T. Ishida, L. Gasser, M. Yokoo: Organization self-design of distributed production systems. IEEE Transactions on Knowledge and Data Engineering 4, n°2, 123-184 (1992).
9. W. Kornfeld, C. Hewitt: The scientific community metaphor. IEEE Transactions on SMC 11, n°1 (1981).
10. A. Kumar, P.S. Ow, M.J. Prietula: Organizational simulation and information systems design: an operations level example. Management Science 39, n°2 (1993).
11. V. Lesser, D. Corkill: Functionnaly Accurate, Cooperative Distributed Systems. IEEE Transactions on SMC 11, n°1, 81-96 (1981).
12 E. Le Strugeon, R. Mandiau, G. Libert: Proposition d'organisation dynamique d'un groupe d'agents en fonction de la tâche. In: Actes des 1eres Journees Nationales sur l'I.A.D. et les Systèmes Multi-Agents. Toulouse, France: 7-8 Avril 1993.
13. E. Le Strugeon, R. Mandiau, P. Millot: Representation of an organization in a multiagent world. To appear in: Proceed. of the IFAC Conference on Integrated systems engineering. Baden-Baden, Germany, 1994.
14. T.W. Malone: Modeling coordination in organizations and markets. Management Science 33, n°10, 1317-1332 (1987).
15. R. Mandiau: Contribution à la modélisation des univers multi-agents: Génération d'un plan partagé. Ph. D. Thesis, L.A.I.H., University of Valenciennes, France, Fev. 1993.
16. C. Numaoka: Conversation for organizational activity. In: Y. Demazeau and J. Muller (eds.): Decentralized Artificial Intelligence, vol. 3. Amsterdam: North-Holland 1992, pp.189197.
17. R.G. Smith: The contract net protocol: high-level communications and control in a distributed problem solver. IEEE Transactions on computers 29, n°12, 1104-1113 (1980).
18. L. Steels: Cooperation between distributed agents through self-organisation. In: Y. Demazeau and J. Muller (eds.): Decentralized Artificial Intelligence, vol. 1. Amsterdam: North-Holland 1990, pp.175-196.

*parc*Plan: a Planning Architecture with Parallel Actions, Resources and Constraints

Jonathan Lever and Barry Richards

IC-Parc, Imperial College, London, SW7 2AZ, UK

Abstract. We describe the generic planning architecture *parc*PLAN, which uses constraint-solving to perform both temporal and non-temporal reasoning. The architecture allows considerable temporal sophistication in the specification of actions, integrity constraints and planning problems, and produces plans with a high degree of concurrency in action execution. A feature of *parc*PLAN is the capability to represent and minimise costs associated with plans. *parc*PLAN has been implemented in the Constraint Logic Programming language ECL^{i}PS^e, and performs well on large-scale planning and resource allocation problems.

1 Introduction

This paper describes the generic planning architecture *parc*PLAN. This architecture is a development of IQ-PLAN [6], and continues to offer the temporal expressiveness of IQ-PLAN while including mechanisms to represent and minimise costs associated with plans, and adding constraint-solving to perform inference.

In planning, actions can be treated as instantaneous—as is typically the case in *state-space planning* (e.g. STRIPS [4])—or can be considered to have duration as in interval-based planning [1]. The *parc*PLAN architecture is interval-based, and is also *non-linear* ([9],[7],[10]) in that the order of action-execution in a plan is independent of the order in which actions are introduced during search.

In plans produced by *parc*PLAN, actions can occur concurrently in a variety of overlapping configurations. The conditions of an action need not be specified as holding *prior* to the start of execution of the action—a more exact specification that involves the precise point during execution of the action at which the condition is required can be given. Similar comments apply to the effects of actions. This allows concurrency of action-execution to be maximised.

An aspect of planning strategies of particular significance for non-linear planning is *least-commitment.* A strategy exemplifies least-commitment if, when faced with a number of alternative choices, it carries forward all possibilities rather than making a (backtrackable) commitment. There are many levels at which choices arise in planning and thus many levels at which the principle may be applied, the most far-reaching being least-commitment on action ordering ([7], [10]). The benefit of least-commitment is in a reduction of choice points, but this must be balanced against the cost of consistency-checking. The strategy used in *parc*PLAN preserves least-commitment on action ordering through part of the planning process, but outputs plans in which actions are totally ordered.

Constraint-solving techniques are relevant to least-commitment as they typically operate by eliminating inconsistent choices without committing to particular values. Constraint-solving has been used to implement least-commitment on the selection of objects used in actions [8], other aspects of resource reasoning [13], and interval durations ([10], [12], [3]). Chapman's TWEAK system [2] applies constraint-solving to temporal and non-temporal aspects of planning, and Yang has described an application of finite domain constraint solving to reason about conflict resolution in planning, taking the TWEAK system as a basis [14]. A planning system that uses temporal constraint propagation in a sophisticated and flexible way was introduced by Allen and Koomen in [1], but the overhead imposed by the temporal reasoner proved to be a bottleneck [5].

In *parc*PLAN, constraint-solving supports least-commitment in many aspects of planning, including resource allocation and action ordering. Temporal reasoning is also achieved through constraint-solving. Time points are represented by finite-domain variables, allowing *parc*PLAN to utilise finite-domain constraint propagation techniques. This commitment to a discrete rather than continuous model of time means that time points cannot be placed arbitrarily close as there is a smallest indivisible time unit between non-simultaneous time points, and that there is a maximum time period that a plan can occupy. However, these factors do not preclude the design of a useful and theoretically sound planner. The fact that the planner operates on a strong underlying temporal representation supports the interval-based, non-linear planning strategy we employ and provides for considerable expressiveness in problem representation.

The *parc*PLAN architecture is *generic*: to apply it to some problem domain it is necessary to provide specifications of actions and integrity constraints. Integrity constraints arise from the problem domain and can be classified as *temporal* or *non-temporal*. Temporal integrity constraints place bounds on the durations of properties and actions, or prohibit certain temporal configurations of properties and/or actions. Non-temporal integrity constraints express other aspects of the problem domain—such as quantitative constraints on resources.

A feature of *parc*PLAN is the capability to represent and minimise costs associated with plans. Such facilities are essential for a planning architecture to have industrial potential, but have been neglected in the planning literature. In *parc*PLAN it is possible, for example, to associate a cost with the performance of a particular action by a particular agent, or to associate a cost with the total quantity of some resource required for the execution of a plan. The architecture includes a number of different cost minimisation strategies that can be utilised within the planning algorithm. The effect of different options can easily be explored by setting global parameters of the architecture, allowing the algorithm to be tuned to particular applications. As a consequence of these features - as well as facilities available in the architecture with respect to temporal reasoning - *parc*PLAN can be seen as a generic shell for solving planning, scheduling and resource management problems involving cost minimisation. In this paper we describe application to blocks-world planning including resources, and to a resource allocation problem provided by British Airways.

2 The *parc*PLAN Architecture

2.1 Representation

Properties and actions holding or occurring over time are represented by terms indexed to time intervals. We use the notation *holds(p,start,end)* to denote that the property or action *p* holds or occurs over the maximal time interval *[start,end)*. The start and end points of time intervals are either integers or finite-domain variables. During planning, the planner works with a database and a set of goals. Both are represented as sets of terms of the form *holds(p,start,end)*.

2.2 Action Specification

Actions are specified by the following:

1. a list of *conditions*
2. a list of *effects*
3. a list of *resource variables*, with associated resource types
4. a list of *constraints*
5. a *cost term*

The conditions and effects of an action are properties or actions indexed to time intervals. The conditions are required for the action to be executed, while the effects are those properties and actions which the action establishes. In particular, a term representing the specified action itself appears among the effects. The resource variables are finite-domain variables that represent resources required in the execution of the action. These variables are associated with a type, according to the type of resource they represent. The constraints are asserted when the action is introduced. The cost term is used in planning to derive the cost of performing the action. The form of the cost term is unspecified by the architecture, and its evaluation is handled by application-specific predicates.

The following specification of the *move* action of the blocks world shows the expressiveness available in *parc*PLAN. The details are discussed below:

conditions:	*holds(on(Block,From),T1,Start),* *holds(clear(To),T2,End),* *holds(clear(Block),T3,T4)*
effects:	*holds(move(Block,From,To,Arm),Start,End),* *holds(on(Block,To),End,T5),* *holds(clear(From),T6,T7)*
resource variables:	*Arm-arm*
constraints:	*designates([Block,From,To],block),* $Block \neq From, Block \neq To,$ $T6 = Start + 2, T2 \leq End - 2,$ $T3 < Start, End < T4, Start + 5 \leq End$
cost term:	*cost(move(Block,From,To,Arm),Start,End)*

Here, the temporal representation offers considerable expressiveness by allowing the point from which the destination block must be clear and that from which the source block becomes clear to be different to the endpoints of the interval *[Start,End)* over which the action occurs. In the above specification the source block becomes clear only after two time units have elapsed from the beginning of the action—perhaps only after the moving block has been lifted sufficiently away from it—through the temporal constraint $T6 = Start + 2$. Similarly, the constraint $T2 \leq End - 2$ requires the destination block to be clear for two time units before the moving block arrives. This kind of flexibility allows maximum interleaving of *move* actions to be achieved by the planner. The final constraint, $Start + 5 \leq End$, forces the action to take at least 5 time units.

2.3 Resource Handling and Costs

For particular applications, resources are specified by a type and quantity. In some cases it is useful to define a notion of equivalence of resources of some particular type, as this can improve the efficiency of some strategies open to the planner by avoiding isomorphic solutions of identical cost.

There are four generic mechanisms that allow costs to be associated with plans. The first is a global notion of cost arising from the total number of resources used in the plan. For particular applications, the cost of using a resource of a each type must be specified. *parc*PLAN sets up delayed calls that monitor the usage of the resource as planning proceeds, and these calls update the value of the cost expression as necessary.

The second notion of cost is local to individual actions, and typically represents the cost of performing a particular action by a particular agent. This is handled by means of the cost term included in the action specification, as described above. The cost term should contain those parameters of the action relevant to the cost, and the evaluation of the cost must be carried out by an application-specific definition. As certain parameters of the action may not have been fixed at the point in planning that the action is introduced, it is often essential to use delay mechanisms in this procedure so that the cost will be re-evaluated as the parameters are filled in.

The third notion of cost is neither local to individual actions nor global to the plan, but is associated with the number of changes in a sequence of values. The fourth notion allows specification of preferred values for variables, and can for example be used to specify a preference that a particular agent carry out a particular action. In both cases, delay mechanisms are used to monitor assignments and update the cost term when necessary.

2.4 Integrity Constraints

Integrity constraints are application-specific and state certain relations between of pairs of properties and/or actions which cannot occur in a valid plan. They are often used to state that two properties cannot hold simultaneously. Integrity constraints for a specific application are defined by clauses for the predicate *integrity_constraint(p,startp,endp,q,startq,endq)*

These clauses are used by the planner to ensure integrity as the plan is built up through the addition of properties and actions. Clauses defining integrity constraints are typically of the form

integrity_constraint(p,Startp,Endp,q,Startq,Endq) :-
blocking_ask test → *action.*

Here, *test* is a constraint involving variables from *p*, *q*, *Startp*, *Endp*, *Startq* or *Endq*, whose truth or falsity is monitored by the generic utility *blocking_ask* supplied by the architecture. If *test* becomes true, *action* is executed, while if *test* becomes false, no action is taken. The action is often to assert a temporal constraint to make the intervals *[Startp,Endp)* and *[Startq,Endq)* disjoint.

2.5 The Planning Algorithm

The planning algorithm can be divided into three phases: action-introduction, collapsing, and labelling of resource and time point variables. The algorithm is broadly similar to that used by Allen and Koomen in [1], although *parc*PLAN goes beyond Allen and Koomen's planner in including metric temporal reasoning and facilities for resource handling.

Before describing the action-introduction phase we must define the notion of potential collapse for a goal with respect to a database:

> A goal *holds(q,startq,endq)* is said to have a *potential collapse* with respect to a database if the goal unifies with an element of the database.

In action-introduction, we select a goal from the current set of goals which has no potential collapse with respect to the current database. Given such a goal, we find an action which achieves the goal, that is, the goal unifies with one of the effects of the action. The action is then introduced by:

1. making the unification between goal and effect
2. adding the effects of the action to the current database
3. adding the conditions to the current goal set and removing the selected goal
4. asserting the constraints
5. evaluating the cost term and adding the result to the current cost

The action-introduction phase then continues with the updated values of database, goals and cost.

When all goals have a potential collapse, *parc*PLAN enters the collapsing phase. The task here is to simultaneously unify all goals with elements of the database. The difficulty is that many combinations of individual collapses are mutually exclusive as they require variables of terms in the database or goals to take incompatible values. In *parc*PLAN, the collapsing phase is handled as a constraint satisfaction problem. Each goal is associated with a finite-domain variable whose domain is a representation of the potential collapses for that goal.

The constraints are that each goal unifies with some element of the database. Choosing a value for a variable chooses the particular element of the database with which the goal associated with the variable is unified. The architecture sets up delayed goals which monitor potential collapses during the choosing of values and implements a looking-ahead strategy over the constraints.

After successfully choosing values for the collapse variables, the database may still contain uninstantiated variables that represent resources or time points. The final phase of planning is to choose values for these remaining variables, whilst minimising the cost function. In fact, in certain variations on the basic planning algorithm that are available in the generic architecture, the choosing of values for collapse variables and of these other variables is interleaved. These various strategies are described in the next section.

If, however, no solution to the collapse exists, the planner must return to the action-introduction phase. At this point, a problem exists as to which goals to regard as unsatisfied and are therefore to be achieved through further actions [1]. This problem is not yet addressed in *parc*PLAN. In order to solve this problem, the failure of the collapse needs to be analysed and a subset of incompatible collapses extracted. The fact that the collapsing phase is handled as a constraint satisfaction problem in *parc*PLAN at least provides a start here, for the use of the first-fail principle in making the collapses leads to early detection of failure.

2.6 Cost Minimisation

The phases of collapsing and labelling resource and time point variables can be subject to a cost minimisation process. The basic algorithm used is branch-and-bound. The best labelling strategy to use is application-specific and *parc*PLAN provides various strategies at the generic level. The ability to compare their behaviour for specific applications by simply setting the value of a global parameter is very valuable. The optimisation strategies currently available are:

1. *length_of_plan*
2. *resource*
3. *eq_resource*
4. *resource_local_cost*
5. *eq_resource_local_cost*

The *length_of_plan* strategy minimises the plan length. The collapse variables are labelled first, then the time points of intervals over which actions occur, and then the remaining variables.

The *resource* strategy minimises the cost function built up during planning as described in Section 2.5 above. Resource variables are labelled first, followed by collapse variables and time point variables. When choosing the value of resource variables, the system prefers values that have already been used in the plan to unused values, helping to obtain a low bound early in the search.

The *eq_resource* strategy is a variation on the *resource* strategy. The values in the domain of the resource being labelled are divided into equivalence classes according to the application-specific definition of equivalence. The idea is that the values in a single equivalence class are interchangeable in a plan without disrupting the overall feasibility of the plan or changing its cost. This notion allows the construction of many isomorphic solutions of identical cost to be avoided in a way that is seen in its most basic form in the graph-colouring example given in [11]. In assigning a value to a resource variable, the values considered from each equivalence class are restricted to those values already used in the plan plus *only one unused value*.

In the *resource_local_cost* strategy, labelling is again initially on the resource variables. When labelling a resource variable, the available values are ranked in order of the increase in the lower bound of the cost function introduced by choosing the value for the variable at this stage of the labelling process. Values that introduce smaller increases in the cost function are preferred.

Finally, the *eq_resource_local_cost* strategy combines the partitioning of resources into equivalence classes with the local cost strategy.

3 Performance

We first describe performance of *parc*PLAN for a formulation of the blocks-world problem including resources. There is a single action, which is to move a block from one position to another using a designated arm. Arms are treated as resources, the number of arms is limited, and one option is to minimise the number of arms needed to execute the plan[1]. Another option is to minimise the length of the plan. We also regard the table as limited to a number of positions on which blocks can be placed.

The following is a randomly generated problem with five blocks and five available arms:

Initial state:	*on(2, 1), on(1, 5), on(5, table1),on(4, 3), on(3, table2)*
Goal state:	*on(1, 2), on(2, 3), on(3, 4), on(4, 5), on(5, table3)*

Optimising on the number of arms required, we obtain the following solution::

move(2, 1, 3, arm2)	over interval *[1, 9]*
move(1, 5, 2, arm1)	over interval *[2, 10]*
move(5, table1, table3, arm3)	over interval *[3, 4]*
move(4, 3, 5, arm3)	over interval *[5, 6]*
move(3, table2, 4, arm3)	over interval *[7, 8]*

The action *move(Block,Source,Destination,Arm)* is read as "move *Block* from *Source* to *Destination* using *Arm*". This solution is found and proved to be optimal with respect to the number of arms required in 1.25 seconds user-time. This is for the ECL^iPS^e implementation running on a Sparc 2.

Setting the optimisation criterion to length of plan, we obtain the following plan (which uses all five arms):

[1] *parc*PLAN minimises the number of arms required for a plan with a minimal number of actions: other plans requiring fewer arms but more actions may exist.

move(2, 1, 3, arm1)	over interval *[1, 7]*
move(1, 5, 2, arm2)	over interval *[2, 8]*
move(5, table1, table3, arm3)	over interval *[3, 4]*
move(4, 3, 5, arm4)	over interval *[4, 5]*
move(3, table2, 4, arm5)	over interval *[5, 6]*

This solution is found and proved to be optimal with respect to plan length in 0.78 seconds user-time.

For ten randomly generated problems involving ten blocks and five arms with optimisation set to minimise the number of arms required, the average user-time to find and prove the optimal plan ranged between 6.17 seconds and 16.12 seconds, the average being 9 seconds. For problems of this scale in which no plan was found due to *parc*PLAN committing to a collapse after introducing insufficient actions, evaluation failed in an average of 2 seconds.

For ten randomly generated problems involving twenty blocks and twenty arms with optimisation set to minimise the length of the plan, the average user-time to find and prove the optimal plan ranged between 6.03 seconds and 10.5 seconds, the average being 7.3 seconds. For problems of this scale in which no plan was found, evaluation failed in an average of 4.4 seconds.

We have also applied *parc*PLAN to a resource allocation problem involving cost minimisation provided by British Airways. The problem has two aspects: to allocate aircraft to flights over the period of a day, and to perform re-allocation in the event of delays during the day. For start-of-the-day allocation, the input consists of:

1. a set of planes, with specification of initial position,time from which available for operation, and operational status of individual aircraft
2. a set of flights, specified by source and destination airports, departure time and arrival time
3. a set of tours - sequences of flights which it is preferred should be carried out by the same aircraft
4. a set of maintenance requests, specifying the aircraft required, and the location and time period of the maintenance

For re-allocation, the input is similar, but includes the current allocation of aircraft to flights, specification of a delay to the arrival-time of a flight, and the time from which re-allocations can be made. In both cases, the output required is an allocation of aircraft to flights that is feasible and which minimises a cost function. In the flight allocation problem, costs can arise in three ways:

1. from the number of aircraft used in the allocation
2. by allocation of an aircraft of special operational status to flights of certain length or to certain destinations
3. by breaking a tour through allocation of different aircraft to adjacent flights

These costs are each assigned a weight and are combined into a single cost function. In re-allocation, changing the aircraft previously allocated to a flight also incurs a cost.

A characteristic of the problem derives from the fact that Heathrow Airport, as the centre of British Airways' operation, is very much the hub of activities. The majority of aircraft start the day at Heathrow, almost all flights either depart from or arrive at Heathrow, and maintenance is carried out at Heathrow. While the allocation to flights from smaller airports is virtually deterministic, the allocation to flights leaving Heathrow, particularly later in the day when there have been many incoming flights, is strongly undetermined. As a consequence, there are many similar allocations that differ only in the choices made for one or two flights, and these allocations tend to have similar cost.

We present timings for start-of-the-day allocation for a data set with sixteen aircraft, forty-one flights and four maintenance requests. Seven of the aircraft are of special operational status. This data set is typical of the problem size and was provided by BA. The timings given are seconds of user-time to find the optimal solution and to the conclusion of the branch-and-bound search proving optimality , using the current ECL^{i}PS^e implementation of the architecture running on a Sparc 2. A "-" in the table indicates that the computation was terminated after 30 minutes without result.

Heuristic	time to optimum	conclusion of proof
none	-	-
local cost	12	-
equivalent resources	44	541
equivalent resources + local cost	7.5	607

The equivalent resources strategy in combination with the local cost heuristic gives the fastest route to the optimal solution. However, the subsequent proof of optimality takes longer than under the equivalent resources strategy alone, as the more complex strategy does more work at each step in the labelling. We feel that there is scope to significantly reduce the time taken up by the proof of optimality by using one strategy to generate a good first solution and a second strategy in the subsequent search for a better solution.

The following table shows the effect of differentiation of resources on the equivalent resources strategy. Timings are given for allocation in the data set used above while varying the number of aircraft with special operational status. In fact, the maintenance requests were omitted from the data as they also effect the equivalence of aircraft. As the number of aircraft of special status increases, so does the number of equivalence classes that must be considered. Timings are user-time in seconds.

	Aircraft with special status					
	0	1	2	3	4	5
time to optimum	6.3	6.4	6.5	6.5	6.5	6.5
conclusion of proof	1.28	1.63	6.67	33.22	81.92	380.02

The local cost heuristic is particularly useful in the case of re-allocation, as introducing a change from the original schedule immediately increases the cost function, so that the values of the original schedule tend to be selected first. For the data set used above and a delay to a flight due to arrive at Heathrow at midday, the optimal repair is found and proved optimal in 3.7 seconds.

4 Conclusions

We have described the generic planning architecture *parc*PLAN, which uses constraint-solving to handle both temporal and non-temporal reasoning tasks. The architecture allows considerable temporal sophistication in the specification of actions, integrity constraints and planning problems, and naturally produces plans with a high degree of concurrency in action execution. *parc*PLAN includes generic mechanisms for cost handling and minimisation. The current CLP implementation of *parc*PLAN is able to solve large-scale planning and resource allocation problems with considerable efficiency.

Acknowledgements This research was carried out on ESPRIT project No. 5291, CHIC. We thank our colleagues on the CHIC project and at IC-Parc.

References

1. J. Allen and J. Koomen. Planning using a temporal world model. In *Proceedings of the 2nd IJCAI*, Karlsruhe, 1983.
2. D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.
3. K. Currie and A. Tate. O-plan: control in the open planning architecture. In *Proc. of the BCS Expert Systems 85 Conf.*, Warwick, 1985. Cambridge University Press.
4. R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.
5. J. C. Hogge. Tplan: a temporal interval-based planner with novel extensions. Technical Report UIUCDCS-R-87-1367, University of Illinois at Urbana-Champaign, September 1987.
6. B. Richards, Y. Jiang, and H. Choi. On interval-based temporal planning - an IQ strategy. In Z. W. Ras and M. Zemankova, editors, *Proc. of the 6th International Symposium ISMIS 91*, pages 226–235, Charlotte, USA, 1991. Springer Verlag.
7. E. D. Sacerdoti. *A structure for plans and behaviour.* American Elsevier, New York, 1977.
8. M. Stefik. Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, 16(2):111–139, 1981.
9. G. J. Sussman. A computer model of skill acquisition. Technical Report AI-TR-297, MIT AI Laboratory Memo, Cambridge, MA, 1973.
10. A. Tate. Generating project networks. In *Proc. of IJCAI-77*, Boston, MA, 1977.
11. P. Van Hentenryck. *Constraint satisfaction in logic programming.* MIT Press, 1989.
12. S. A. Vere. Splicing plans to achieve misordered goals. In *Proc. of IJCAI-85*, pages 1016–1021, 1985.
13. D. E. Wilkins. Domain independent planning: representation and plan generation. *Artificial Intelligence*, 22:269–301, 1984.
14. Q. Yang. A theory of conflict resolution in planning. *Artificial Intelligence*, 58:361–392, 1992.

GAITS II : An Intelligent System for Computer-Aided Education

Mohamed QUAFAFOU

IRIN, Université de Nantes, 2 Rue de la Houssinière
44072, Nantes Cedex 03, France.
quafafou@irin.univ-nantes.fr

Abstract. The design of computer-aided education systems is an important area that offers many challenges to the artificial intelligence community. The problems involved in such design are ambiguous and the managed data are imprecise. The coupling of symbolic and numerical computing might be the key for making educational environments more robust, and better in handling the involved problems. This paper presents mechanisms for coupling symbolic and numerical computing in intelligent educational environment. The proposed method combines heuristic AI methods with analytical control of adapting computer-aided education. A coupled system (GAITS II) with symbolic and numerical frameworks is described in this paper. A rule-based inference engine for diagnostic reasoning is coupled with an optimization method to select an appropriate dialogue (prototype remediation) for a student. The system deals with both the symbolic and numerical methods using not only an inference engine but also fuzzy sets theory and optimization methods (genetic algorithms). We present a new optimization method to select an appropriate dialogue for a student defining a new global mapping function between the pedagogical guidelines and the corresponding numerical terms.

1 Introduction

The need to have a computer understand its user and make intelligent decisions during their interactions is a hallmark of problems in AI. A lot of work has been done in the AI community to tackle the problem of integrating intelligent computers and human users into an effective cooperative system [1]. However, the use of interfaces to support a particular style of interaction is not sufficient. The coupling between the human and intelligent system must be an integral part of the design of intelligent systems.

Intelligent tutoring research considers a special type of user, which is the student. Intelligent Tutoring Systems (ITSs) are designed to produce in a computer beviour which, if performed by a human, would be described as "good teaching" [15, 16]. Such systems can make intelligent decisions to deliver effective remedies considering the relevant information about the student [2]. The first generation systems inhereted the IA symbolic approach considering only a symbolic processing of the information. Unfortunately, an ITS handles different knowledge (i.e., numerical knowledge, heuristics and symbolical concepts) to achieve the pedagogical goal. In this paper we describe a coupled educational system based on two main components: the diagnostic component, and the tutorial component. The diagnosis process based on an inference engine provides the appropriate pedagogical guidelines to control the next teaching session. An objective function is then defined representing a numerical model of the provided guidelines. Then, the teaching session is based on the dialogue resulting from the optimization of this function. This system is designed to manage both symbolic and numerical knowledge to improve the decision process within tutoring environments.

2 Symbolic and numerical computing with tutoring systems

2.1 A brief introduction to the coupling of symbolic and numerical computing

The coupling of symbolic and numerical methods is the integration of formal mathematical methods with methods based on symbolic knowledge. This coupling has become an area of growing interest and might be the key for making problem solving environments more robust, and better in handling problems involving ambiguous, contradictory, or imprecise data. Kitzmiller and Kowalik [3] have presented a discussion of several research issues focusing the discussion around the definition of coupled systems, motivations for coupling, coupled system architecture, and key factors in the design of coupled systems. Many systems have been developed in search of a good methodology of this coupling [4, 5].Two basic types of systems have been defined and referred to as *shallow* and *deep* coupled systems. Shallow coupling means that the coupled system tends to treat mathematical models as black boxes and have little knowledge of involved processes. Such systems can be used to help the user to apply numerical algorithms to the task at hand and to interpret the result of numerical routines. The deep coupled systems extensively use knowledge of the processes involved, and exploit the data structure. Such systems have been used in control systems design environments.

2.2 Why is it important ?

In recent years, many companies are actively pursing means to integrate educational systems in its development. An important result of the use of educational software in industry is the increase of the engineer's labor productivity and a considerable gain of time. It may also have a significant impact on both education and acceleration of technological progress. However, the real world processes to be taught are complex to formulate and generally imperfectly known [6, 7]. The coupling of simulators in educational systems correspond to real needs allowing the simulation of impossible or dangerous scenery and the reduction of the training time necessary in the real world.

A compute-aided education system provides a suitable environment for discussing the problem of coupled systems, where not only the numerical models of phenomena to be taught are to consider, but also both the user's characteristics and pedagogical consi*derations* are of primary concern. Different processes must be coupled to improve both the *adaptivity* (ability of the system to adapt its behavior according to their degree of adaptivity to the user's behavior) and the *reactivity* (capability of the system to evaluate and execute a set of functions available to the user) of the educational system. The emerging new technologies in computers, networking and distance learning make this topic important. The importance of the research here is twofold:

- The gist of the current problem is an issue of developing a hybrid environment supporting symbolic and numerical computing and coupling rule-based methods, genetic algorithm-based optimization technique and fuzzy logic-based machines. This issue is fundamental to the study of education and training systems, which must be used in real world (i.e., industrial processes, air training command and technical education and training).
- Coupled education systems promise to integrate the explanation and problem solving capabilities with the precision of traditional numerical computing. The current researches are great potentials in developing computational models for teaching and training.

3. Towards a coupled computer-aided education system

The AI community has been participating actively to the development of new educational systems integrating latest development in AI architecture and machine learning [8, 9]. In the principle, The ITSs inherited the AI symbolic approach. Therefore the use of expert systems as an educational tool was a principal problem. The advantages of this approach are the availability of a knowledge base gathering the human expert knowledge and the explanation facility. The knowledge collected is then transferred to the student to acquire the knowledge and the feeling necessary to deal with complex problems. On the other hand, the simulation constitutes a powerful tool to help analyze of complex problems. O'Keefe [10] has presented four models for combining simulation and expert systems. These two technologies can be usefully combine in instructional applications allowing the interpretation of complex simulation results. The simulation capabilities make the expert system able to look into the future before reasoning [11, 12]. For instance, Regian and Pitts [13] have constructed a tutor based on "fuzzy relations" approach for Air Training Command.

Our objective is to design an ITS to manage an adapted instruction in the sense that the level of teaching vary as the student performances evolve taking into account of the pedagogical guidelines. The main strategy adopted to assume a tailored interaction is pragmatic and easy to understand. The system works in both a symbolic and a numerical universe. An inference engine allows to deduce the appropriate guidelines for a given student before starting each teaching session. Then, these guidelines are translated to numerical terms and an objective function is constructed. The result of the function minimization is an optimal dialogue, which guide the selection process of a real dialogue. This later is define by the author and it will be at the basis of the interaction in the next teaching session. In the symbolic universe, the traces of the inferences can be used to explain the teaching strategy adopted by the tutor, when in the numerical universe, the research in a continuous space of the suitable dialogue uses the objective function. The integration of these complementary methods leads to the construction of intelligent systems managing in an effective way symbolic and numerical information.

4. Architectural considerations

4.1 The coupled processes

Previously, Quafafou [14] has developed a prototype called GAITS, based on a supervised teaching paradigm. First, the author establishes dialogues by making the inventory of the teaching content. Before starting the teaching session, the teacher assigns a "teaching interval" and a "session teaching control function" to each student. The teaching interval defines the initial profile (the student knowledge presupposed model) and the final profile (the final state where the teaching objective is judge to be reach). The control of the interaction with the student is base on the "sessions teaching control function" that is an analytical model represented by an objective function. This function is to optimized at each teaching session to find the optimal profile to met at this session. Three principal processes compose GAITS (Figure 1-a):

- *Admissible dialogues selection process (Process 1)* : It selects a subset of dialogues from the dialogues base. These dialogues are called admissible dialogues (their prerequisites are satisfied) and define the session context.
- *Appropriate dialogue selection process (Process 2)* :It selects from the session context the most appropriate dialogue to the student.
- *Teaching process (Process 3)* :It interacts with the student considering this selected appropriate dialogue.
- *Appropriate dialogue selection process (Process 2)* : It selects from the session context the most appropriate dialogue to the student.
- *Teaching process (Process 3)* : It interacts with the student considering the selected appropriate dialogue.

The Process 1 and Process 2 manage numerical knowledge applying operators to fuzzy sets. They allow to choice the appropriate dialogue using the objective function, which represents a numerical model of pedagogical guidelines. The system updates the student model at the end of a teaching session (the end of the process 3) and stops if it met the pedagogical objective. In the contrary, it restarts a new session using the same objective function. However, the pedagogical guidelines used are static. Two processes are added, allowing a dynamic generation of the appropriate guidelines, to improve the adaptivity of the system. In other word, the objective function is not static, but it is dynamically construct before starting any teaching session (Figure 1-b):

- *the pedagogical guidelines inference process (Process 4)* : At the end of each teaching session the system diagnoses the student situation and infers the appropriate guidelines to control the next teaching session.
- *the translation process (Process 5)* : It translates the appropriate guidelines inferred to numerical terms and a numerical model (the objective function) is constructed.

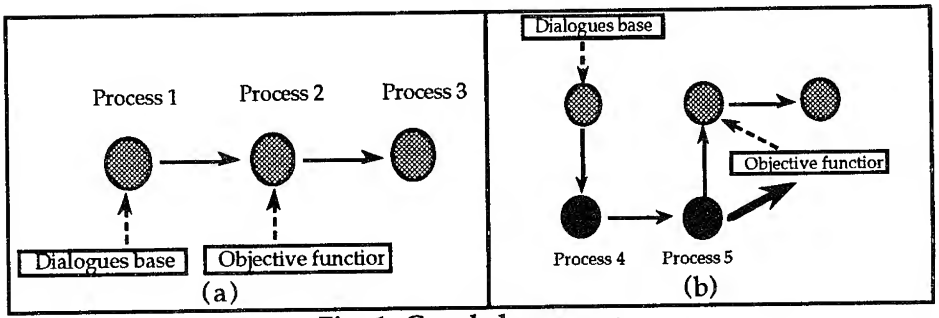


Fig. 1. Coupled processes

The process 4, discussed in Section 5, is based on a diagnosis inference engine. Each inferred guideline is then translate by the process 5 into a numerical constraint. The Section 6 gives more information on this process.

4.2 The general architecture

The first prototype uses simple coupling scheme and considers few processes. The components are used as black boxes, in a shallow way, with a clear separation of symbolic and numeric functions. They communicate through a global memory and the student model defining two symbolic and numerical frameworks (Figure 3). First, the selection of admissible dialogues (session context) process computes a degree of acceptance, used in the rest of the session, for each dialogue. The dialogues that deviate from the educational strategy are eliminated. Then, the diagnostic reasoning process infers the student's appropriate guidelines considering both the student model and the pedagogical rules. These rules describe mechanisms that a human teacher use to guide the student tacking into account of his level. The pedagogical guidelines represent symbolic terms as "select only dialogues teaching doubtful concepts", "consider the previous objective" and "the evolution of the student's knowledge must be feeble". The result of the diagnosis reasoning process is a set of guidelines connected by logical operators (e.g., and ($\wedge$) , or ($\vee$) , not ($\neg$)).

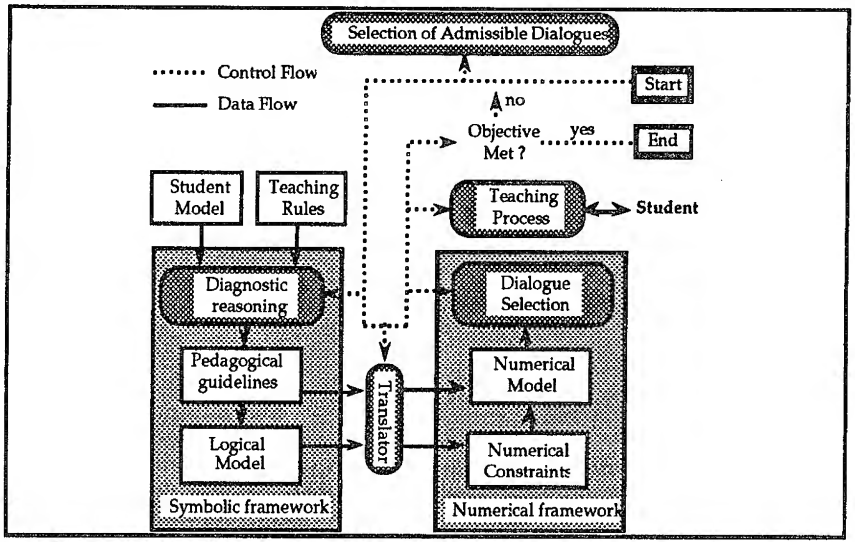


Fig. 2. Architecture with symbolic and numerical frameworks

The role of the translator is to traduce the symbolic information (guidelines) inferred by the diagnostic process to numerical terms, which are at the basis of the construction of the objective function. This function is a numerical model of the pedagogical guidelines. The optimal objective to met at the next session is the result of the optimization of the objective function. This optimal objective allows to select an "actual dialogue" from the session context. Afterwards, the system interacts with the student using the actual dialogue. At the end of this interaction the student model is updated and performance degrees, useful for the teaching diagnosis, are computed. The system stops if the pedagogical objective is reach, otherwise a new session starts. In this context, any unnecessary knowledge, from desired target point of view, must not be presented to the student. So, the system continuously supports the student to reach rapidly the expected educational target.

5 An inference engine for diagnostic reasoning

5.1 The knowledge base

At the end of a teaching session (process 3), the system updates the student model and verifies whether the pedagogical objective is met. When the pedagogical objective is not reach a new session restarts and a new context (admissible dialogues) is defined. Then pedagogical rules, used to infer the appropriate guidelines, represent the knowledge that allows to a teacher to guide the student controlling his knowledge evolution process, his performances and errors. They manipulate knowledge linked to *the student level* (represent the current state of the student's knowledge), *the acquired knowledge* (concepts mastered by the student), *the doubtful knowledge* (we are not sure that this knowledge is mastered) and *the objectives* , which are goals that drive the teaching session. The following rules are like examples of pedagogical rules:

R_1 : **If** Average_Result is high **and** average evolution **then** *maximum evolution* **and** *finale Objective*

R_2 : **If** Doubtful_Concepts **then** Select dialogues linked to the doubtful concepts **or** *feeble evolution*

R_3 : **If** Average_Result is high **and** Current_Result is feeble **then** Accident **and** Select dialogues teaching concepts belonging to both the previous dialogue and to the finale profile

R_4 : **If** Average_Result is law **then** feeble_student

R_6 : **If** Accident **then** *previous Objective* **or** *feeble evolution*

R_5 : **If** feeble_Student **then** Select dialogues teaching concepts imperfectly known

The diagnostic reasoning process consists in the deduction of conclusions related to the student's knowledge evolution, the objective to consider, the knowledge to take into account and concepts to present. The deduced knowledge plays an important role in developing a more adapted instruction providing versatility and sensitivity to students.

5.2 Control variables and pedagogical guidelines

The knowledge base contains rules that consider different control variables (e.g., End_Session, feeble_Student and Doubtful_Concepts) updated at the end of each teaching session. This base allows to infer the appropriate guidelines for given student at the start of the current teaching session. A global mapping function between the pedagogical guidelines and corresponding numerical terms is define by the author. The evolution of the student knowledge can be defined as the distance between the current state of the student knowledge (Current_Profile) and the state that will be reached at the end of the next teaching session. So, the optimal dialogue η_T we have to find must be as closed as possible to Current_Profile. Consequently, the guideline "*feeble evolution*" is traduce by the numerical constraint min d(η_T, Current_Profile). In other hand, if the diagnosis process infers "*finale Objective* " then the optimal profile η_T to find must be as closed as possible to the finale objective. Consequently, we minimize the distance d(η_T, Finale_Profile).The Table1 gives examples of mapped terms.

Table 1 Mapping between guidelines & numerical constraints

Symbolic	Numeric
finale Objective	min d(η_T, Finale_Profile)
previous Objective	min d(η_T,η_{T-1})
feeble evolution	min d(η_T, Current_Profile)
maximum evolution	max d(η_T, Current_Profile)

Remark : The mapping function (defined by the author) is use because the system has little knowledge of the involved processes. This shallow way allows us to develop rapidly an initial prototype. It is only a first step in developing a coupled system.

5.3 Rules with actions

The conclusion of a rule can contain also an action as "Select", which is defined by a function that computes a kind of utility degree for each candidate dialogue, knowledge or concept by aggregating several criteria. The result at the end of the diagnostic reasoning process is a table containing utility degrees (computed by the action Select) as :

Table 2 Utility degrees of dialogues

Rules	Dialogues					
	D_1	D_2	D_3	D_4	D_5	D_6
R_2	.3	0	0	0	0	1
R_3	0	0	.3	0	1	.6
R_5	.5	.4	0	1	0	.8

The rule R_2 indicates that dialogue D_6 teaches all doubtful concepts and the absence of these concepts in the dialogues D_2, D_3, D_4 and D_5. Each rule with an action infer a numerical terms defined by both the utility degree w_i and distances between the optimal dialogue and the admissible dialogues as :

$$\frac{1}{\sum_{i=1}^{k} w_i} * \sum_{i=1}^{k} w_i * d(\eta_T, D_i)$$

Consequently, the four rules shown in the table 1 deduce the following numerical expression:

$$R_2 \quad : \quad \frac{1}{1.3} * (.3 * d(\eta_T, D_1) + d(\eta_T, D_6))$$

$$R_3 \quad : \quad \frac{1}{1.9} * (.3 * d(\eta_T, D_3) + d(\eta_T, D_5) + .6 * d(\eta_T, D_6))$$

$$R_5 \quad : \quad \frac{1}{2.7} * (.5 * d(\eta_T, D_1) + .4 * d(\eta_T, D_2) + d(\eta_T, D_4) + .8 * d(\eta_T, D_6))$$

At the end of the diagnostic reasoning process the numerical constraints inferred by rules using actions are added to those generated by translation of the pedagogical guidelines. They are then use to construct the numerical model. The task of the optimization module is to determine the fuzzy set η_T that defines the optimal profile.

6 The numerical model construction

The system chooses the appropriate guidelines taking into account of the pedagogical guidelines inferred at the beginning of each teaching session. The result of the diagnosis process is a set of guidelines and functions, which must be execute. Each inferred pedagogical guideline is then substitute by its corresponding numerical term and the results of the functions are numerical constraints. Then the objective function is constructed. Suppose that at the end of a session the average result of the student is law and there are some doubtful concepts. Using the five precedent rules the system infers A (Select dialogues linked to the doubtful concepts), B (*feeble evolution*) and C (Select dialogues teaching concepts imperfectly known). The logical model (A $\vee$ B) $\wedge$ C) is translated and the result is the numerical model (A' * B') + C' (Figure 4) :

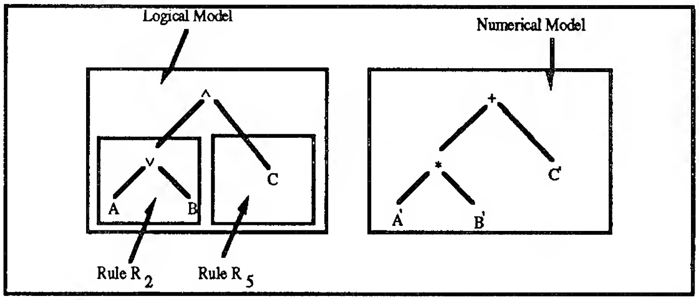


Fig. 3 A simple example of corresponding models

where $A' = \frac{1}{1.3} * (.3 * d(\eta_T,D_1) + d(\eta_T,D_6))$

$B' = d(\eta_T, \text{Current\_Profile})$

$$C' = \frac{1}{2.7} * (.5 * d(\eta_T,D_1) + .4 * d(\eta_T,D_2) + d(\eta_T,D_4) + .8 * d(\eta_T,D_6))$$

Note that the semantics of "d" a fuzzy sets distance depends on the pedagogic rule considered. For example d(η_T, Current_Profile) denotes the student progression knowledge.

7 Conclusion

This paper offers mechanisms for coupling symbolic and numerical computing in intelligent tutoring environment. We propose a method to combine the worlds of qualitative and heuristic artificial intelligence with the quantitative and analytical control of an adapting tutoring. The system developed integrate the explanation and problem solving capabilities with the precision of numerical model for more adapted instruction providing versatility and sensitivity to students. Such approach can probably play an important role in developing high adaptive instruction. Recently, we have integrated a symbolic method to learn rules from all contexts in which the student was placed and acquired his knowledge. The learned rules support the making decision process and allow a dynamic generation of hypotheses for more effective and adaptive interaction. Besides, we are developing a "goal-driven explanation process" which refers to the process of using goals of an intelligent tutoring system to make decision about when to explain, what should be explained, and which explanation strategies are appropriate for a given student and a context. The approach outlined in this paper is still incomplete for the fusion of the two technologies. We wish to remark that pedagogical and technical issues remain to be addressed and we are currently dealing essentially with the following problems : how the adapt dynamically the pedagogical rules ? and how to perform the explainability of the tutor ?

References

1. D.D. Woods, L. Johannsen, S. S. Potter : Human Interaction with Intelligent Systems: An Overview and Bibliography SIGART Bulletin,Vol. 2,No 5.
2. E. Wenger : Artificial intelligence and tutoring systems, computational and cognitive approaches of the communication of knowledge. Los Atos, Morgan Kaufman Publishers inc., 1987.
3. Kitzmiller, C.T. and Kowalik, J.S. Symbolic and Numerical Computing in Knowledge-Based Systems, Coupling Symbolic and Numerical Computing in Expert Systems, Elsevier Science Publishers, pp. 3-17, 1986.
4. Simmons, M.K., Dixon, J.R. Reasoning about Quantitative Methods in Engineering Design, Coupling Symbolic and Numerical Computing in Expert Systems, Elsevier Science Publishers, pp. 47-57, 1986.
5. Furuta, K. and Smithers, T. Numerical Methods in AI-Based Design Systems, Applications of Artificial Intelligence in Engineering VI, Computational Mechanics Publications & Elsevier Applied Science, pp. 45-58, 1991.
6. M. Quafafou, J.P. Rolley, O. Dubant, P. Prévôt The design of a simulator system for educating engineers. Proceeding of The Sixth international conference on Artificial Intelligence in Engineering , Oxford, July 2-4, 1991, p. 435-454.
7. M. Quafafou, P. Prévôt The design of an ITS dedicated to industrial proceedings IEEE International Conference on Developing & Managing Intelligent System Projects, Washington (USA), March 29-31,1993.
8. Cumming, G., Self, J. : Collaborative Intelligent Educational Systems. In Bierman, D., Breuker, J., and Sandberg, J. (Eds) Artificial Intelligence and Education : Synthesis and Reflection. IOS, Amsterdam, The Netherlands, 1989.
9. Hoskin, N., Aiken, R. M. : A machine-learning compagnion. In proceeding of the IFIP 12th world computer compress, Madrid, September 1992, 188-196.
10. O'Keefe, R.M. : Simulation and expert systems - A taxonomy and some examples. Simulation, January 1986, Vol. 46, No 24, pp. 77-98.
11. Lounamaa, P. and Tse, E. The Simulation and Expert Environment, Coupling Symbolic and Numerical Computing in Expert Systems, Elsevier Science Publishers, pp. 83-99, 1986.
12. Hacid, M. S., Bonnet, C., Kouloumdjian, J. : Knowledge-based simulation in memory re-education. In proceeding of the IFIP 12th world computer compress, Madrid, September 1992, 217-223.
13. Regian, W., Pitts, G. : A fuzzy logic-based intelligent tutoring system. In proceeding of the IFIP 12th world computer compress, Madrid, September 1992, 66-72.
14. Quafafou, M., Nafia, M. GAITS : Fuzzy Sets-Based Algorithms for Computing Strategies using Genetic Algorithms, 8th Austrain Artificial Intelligence Conference, FLAI'93, Linz, Austria, pp.59-67, June 1993.
15. Hyacinth S. Nwana. Intelligent Tutoring Systems : an overview. Artificial Intelligence Review, (1990) 4, 251-277.
16. Elsom-Cook, M. Intelligent Computer-Aided instruction research at the Open University. Technical Report No 63. Computer-Assisted Learning Research Group, The Open University, Milton Keynes.

The GLS Discovery System: Its Goal, Architecture and Current Results

Ning Zhong   Setsuo Ohsuga

Research Center for Advanced Science and Technology
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153, Japan

Abstract. We have been developing a system called GLS (Global Learning Scheme) for knowledge discovery in databases. The development of GLS has two main aspects. The first is to develop a *multi-strategy* system. That is, many kinds of discovery/learning methods are integratedly used in multiple learning phases for performing multi-aspect intelligent data analysis as well as multi-level conceptual abstraction and learning. As a multi-strategy system, GLS is implemented as a toolkit that is composed of several sub-systems and optional parts with multi-level structure. We have finished main parts belong to this aspect, and have undertaken another aspect, i.e., extending GLS into a *multi-agents, distributing* and *cooperating* discovery system. We try to increase autonomy of discovery process by increasing the number of discovery steps in succession performed in both the centralized and distributed cooperative mode. This paper briefly describes the GLS system: its goal, architecture and initial implementation, and discusses further research directions.

1 Introduction

Knowledge discovery in databases (KDD) is becoming an important topic in AI and is attracting the attention of leading researchers in databases [9]. Several discovery systems such as INLEN, KDW and Forty-Niner (49er) etc. have been developing [3, 10, 19]. This topic draws attention from several fields including expert systems, machine learning, intelligent databases, knowledge acquisition, case-based reasoning and statistics. Thus, this topic is different from traditional researches of machine learning, though it uses their results [4, 9, 11]. We have been developing a system called GLS (Global Learning Scheme) for knowledge discovery in databases [13]. The development of GLS has two main aspects.

The first is to develop a *multi-strategy* system. Since databases have the following features different from other learning objects such as (1) databases are not always complete but contain uncertain and incomplete data; (2) there are different kinds of data such as numerical data and symbolic data in databases; (3) databases are generally very large and complex; (4) databases for discovering knowledge are not always static but dynamic, we cannot wish a single discovery/learning algorithm for solving all problems. To meet the features of databases, we adopt the process of discovering knowledge from databases based on incipient hypothesis generation, evaluation and refinement in GLS as shown in Fig.1 [8]. In this process, many kinds of discovery/learning methods are integratedly used in multiple learning phases for performing multi-aspect intelligent data analysis as well as multi-level conceptual abstraction and learning.

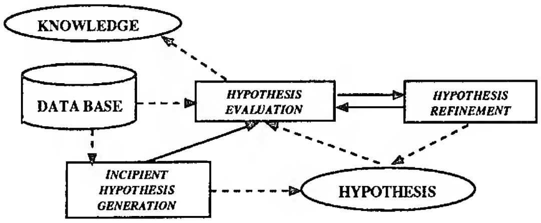


Fig. 1. The process of knowledge discovery from databases

As a multi-strategy system, GLS is implemented as a toolkit that is composed of several sub-systems and optional parts with multi-level structure. We have finished main parts belong to the multi-strategy aspect, i.e., two sub-systems for incipient hypothesis generation, DBI (Decomposition Based Induction) and KOSI (Knowledge Oriented Statistic Inference), and two sub-systems for refinement/management, HML (Hierarchical Model Learning) and IIBR (Inheritance Inference Based Refinement), have been implemented [16, 15, 17, 14, 18]. Furthermore, we have undertaken another aspect, i.e., extending GLS into a *multi-agents, distributing* and *cooperating* discovery system. We try to increase autonomy of discovery process by increasing the number of discovery steps in succession performed in both the centralized and distributed cooperative mode. GLS is implemented by KAUS. KAUS is a knowledge-based system developed in our laboratory which involves knowledge-bases based on MLL (Multi-Layer Logic) and databases based on the NNF (Non Normal Form) model [6]. Thanks to the useful capabilities such as meta reasoning, multiple knowledge worlds and the model representation in KAUS, and the development of distributed KAUS (DKAUS) recently [7, 12], GLS can be easily implemented and extended by KAUS.

This paper briefly presents the GLS discovery system: its goal, architecture and current results. It includes to describe briefly a *global learning scheme* that is the basic framework of developing the GLS system, discuss an experimental application, describe how to extend GLS into a *multi-agents, distributing* and *cooperating* discovery system, and give other future research subjects.

2 Global Learning Scheme

Global learning scheme as shown in Fig.2 is the basic framework of developing the GLS system. This is a very general scheme that serves the process of knowledge discovery from databases shown in Fig.1. By means of it, multi-aspect intelligent data analysis as well as multi-level conceptual abstraction and learning can be easily implemented by multiple learning phases in an environment of integrated use of knowledge-bases and databases. This scheme is mainly divided into three phases as shown in Fig.2:

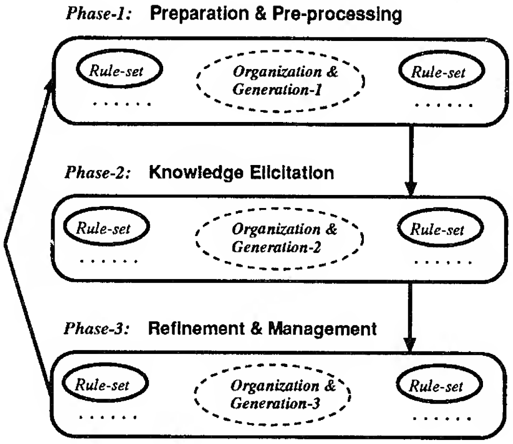

Phase-1: Preparation & Pre-processing
Rule-set
Organization & Generation-1
Rule-set
Phase-2: Knowledge Elicitation
Rule-set
Organization & Generation-2
Rule-set
Phase-3: Refinement & Management
Rule-set
Organization & Generation-3
Rule-set

Fig. 2. Global learning scheme

- Phase 1 is *preparation & pre-processing.* One of the roles of this phase is to decide automatically or interactively the discovery methods to be used, the database name and the attributes that are expected to be effective for knowledge discovery. Sample data are also collected and pre-processed in this phase. For example, *attribute oriented clustering* can be used for quantizing continuous domains, and eventually perform conceptual abstraction (generalization) or *attribute calculation* can be used for generating new attributes in a database [16, 15, 17]. These methods for pre-processing are optional parts in GLS.
- Phase 2 is *knowledge elicitation.* In this phase, eliciting knowledge from databases is done by the discovery method selected in Phase 1. Some discovery methods such as DBI and KOSI, which are the sub-systems of GLS, can be used [16, 17]. As a result, an incipient hypothesis is generated.
- Phase 3 is *refinement & management.* The main steps in this phase are to represent the discovered hypothesis in Phase 2 as the MLL (Multi-Layer Logic) formula in a knowledge-base, and refine/manage the incipient hypothesis by using the methods based on meta reasoning and multiple worlds, domain knowledge, and/or along with the change of data in a database. Some methods such as HML and IIBR, which are the sub-systems of GLS, can be

used for this purpose [14, 18]. When new data is introduced or the acquired knowledge is not sufficient for a given application, the process goes back to Phase 1 for making further discovery.

As shown in Fig.2, in Phases 1, 2 and 3 of this scheme, there are the parts called *organization & generation.* Self organization is necessary for organizing dynamically more discovery steps in succession performed in either the centralized or distributed cooperative mode according to different discovery tasks. The GLS system itself based on this scheme is a knowledge-based system. However, some knowledge (rules) in GLS cannot be given in advance, but must be generated at the discovery time. For example, some rules related to retrieving databases, data focusing and pre-processing must be generated dynamically either through dialogue with the user or by the automatic learning capability. On the other hand, the knowledge discovered from databases must be also represented as the deductive rules and added in a knowledge-base, and can be updated for refinement and management.

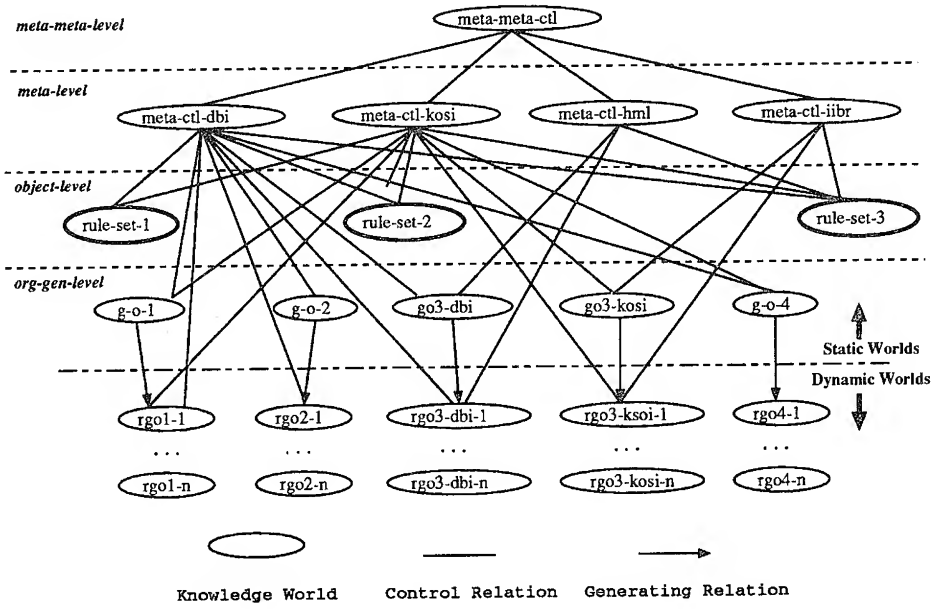


Fig. 3. Hierarchical structure of multiple worlds in GLS

Based on the *global learning scheme* as stated above, the architecture of the GLS system is developed by multiple meta-levels & worlds as shown in Fig.3. That is, all knowledge in GLS is managed by multiple worlds and is divided into four levels (i.e., the meta-meta control level, the meta control level, the object level and the organization-generation level) with hierarchical structure. The functions of the meta (or meta-meta) control level are to control the reasoning for the knowledge worlds of the object (or meta) level. They select the knowledge worlds that should be used, and segregate, organize, modify and use them, so that the result can be obtained as fast as possible. Also, they acquire and manage the knowledge in the object (or meta) level. That is, they generate knowledge worlds dynamically, check their consistency and refine them. In other words, in GLS, learning is not only to discover the knowledge according to the user requirement (we call *object-level learning*), but also to organize dynamically the discovery process according to different discovery tasks and improve the performance of GLS itself (we call *meta-level learning*).

3 An Experimental Application

We have tested or are testing several databases in the fields such as economics, zoology, medicine and material science. In this section, we briefly discuss an example of experimental applications for further describing GLS. It involves to discover concept clusters from a breast cancer database by DBI, and the discovered concept clusters are represented as the *classification knowledge* with hierarchical models in a knowledge-base, and is refined/managed by HML. Since to describe each of sub-systems of GLS in detail requires much more space than we have in this paper, we only describe the process of discovery, refinement and management based on the *global learning scheme* shown in Fig.2. The details on DBI and HML refer to [16, 14].

In this breast cancer database, each tuple corresponds to one patient and values of 10 attributes are given for each patient. The domain of every attribute is given by the sets of 9 quantized values that are classified as a case of benign or malignant cancer, resulting from clinical examinations related to this disease [2]. The meanings of the attributes used in Table 1 are as follows:

Table 1. DB: breast-cancer

id	code-n	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9
1	160296	4	5	8	8	10	5	10	8	10	3
2	342245	2	1	1	3	1	2	1	1	1	1
3	428598	2	1	1	3	1	1	1	2	1	1
4	492561	2	4	3	2	1	3	1	2	1	1
5	493452	2	1	1	3	1	2	1	1	1	1
6	493452	2	4	1	2	1	2	1	2	1	1
7	521441	2	5	1	1	2	2	1	2	1	1
8	560680	2	3	1	2	1	2	1	2	1	1
9	636437	2	1	1	1	1	2	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...

code-n - code-number (Sample Code Number)
a0 - b-cancer-type (Breast Cancer Type: 2 for benign, 4 for malignant)
a1 - clump-t (Clump Thickness)
a2 - u-cell-size (Uniformity of Cell Size)
a3 - u-cell-shape (Uniformity of Cell Shape)
a4 - marginal-adhesion (Marginal Adhesion)
a5 - s-e-cell-size (Single Epithelial Cell Size)
a6 - bare-nuclei (Bare Nuclei)
a7 - bland-chromatin (Bland Chromatin)
a8 - normal-nucleoli (Normal Nucleoli)
a9 - mitoses (Mitoses).

In order to describe the change of data in a database, the breast cancer database is divided into two groups: *group* 1 for fundamental data and *group* 2 for its variation. The objective is to find which conditions of the 9 attributes indicate malignant and which no malignant cancer, and form concept clusters by decomposing this database, so that finally to represent the results as the *classification knowledge* with hierarchical models in a knowledge-base and refine/manage them. That is, this discovery process is divided into two main stages.

The **first stage** is to decompose a database for forming concept clusters by the following main steps in DBI [16]:

*step*1 : Create a *Probability Distribution Matrix (PDM).* There are many kinds of methods for creating the PDM, depending on their purposes. For our application, the dependency relations between any two attributes are considered, their probability distributions are calculated and recorded in a PDM. Let $\mathbf{a} = \{a_1, a_2, ..., a_n\}$ and $\mathbf{b} = \{b_1, b_2, ..., b_m\}$ be the sets of different values of any two attributes in a database that has been preprocessed. Using conditional probability, we have

$$p(x_i|x_j) = \frac{p(x_i \cap x_j)}{p(x_j)} \qquad x_i, x_j \in \mathbf{a}, \mathbf{b}. \tag{1}$$

From this we define p_{ij}, the probability distributions, to be $p(x_i|x_j)/N$, where N is the number of attributes. These p_{ij} constitute the entities of the PDM.

*step*2 : Form the diagonal matrix. It is a step as pre-processing before decomposing the PDM. Two methods, *diagonalization by a special attribute* as a supervised method and *diagonalization by the optimum decomposition* as an unsupervised method, can be used for this according to the cases in which a criterion of forming the diagonal PDM is given by the user or not. Since there is obviously a special attribute (i.e., the attribute *b-cancer-type*, or e.g., c_1, c_2 as shown in Fig.4) that can be chosen by the user as a criterion of forming the diagonal PDM for this breast cancer database, the method, *diagonalization by a special attribute*, is used for obtaining the diagonal PDM shown in Fig.4.

*step*3 : Decompose the diagonal PDM by a decomposing algorithm. In decomposing, primary factors for describing some concepts are aggregated by selecting proper attributes, and two kinds of noises, *minor elements* and *irrelevant elements* are neglected. Where, the *minor elements* are those whose probability

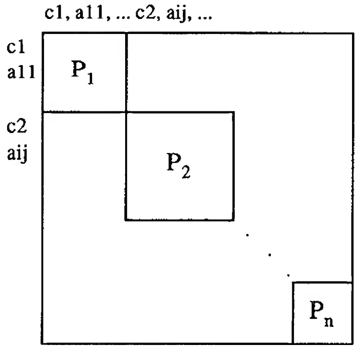


Fig. 4. Forming the diagonal PDM by a special attribute

values are much smaller than other related values. For example, in the *breast cancer database,* if bare-nuclei = 1 then the probabilities of malignant and benign are 0.02 and 0.98 respectively. Thus bare-nuclei = 1 cannot be used as one of the conditions of malignant cancer. Therefore, it is neglected as a minor factor. The *irrelevant elements* are those who cannot be used to differentiate concepts. For example, in the *breast cancer database*, if mitoses = 3 then the probabilities of malignant and benign are both nearly 0.5, even though some data are not appearing in Table 1. Thus mitoses = 3 cannot be used to differentiate malignant from benign. That is, mitoses = 3 is useless for knowledge discovery and cannot be classified into a cluster. Therefore, they should also be omitted as an irrelevant element although its probability value may be fairly large. As the result of decomposing a PDM, several sub-matrices are formed. These results are applied back to a database that has been preprocessed and the job of decomposing the database is thus completed. As the result of decomposing the database, concept clusters are formed.

For example, the following two concept clusters can be discovered from *group* 1 of the breast cancer database:

1. The conditions of benign cancer:
   clump-t-2: 1, 4, 2, 6.
   u-cell-size-2: 1, 3, 2, 9.
   u-cell-shape-2: 2, 1, 4.
   s-e-cell-size-2: 1.
   bare-nuclei-2: 0, 1, 3, 8.
   bland-chromatin-2: 1, 3, 4.
   mitoses-2: 2.
2. The conditions of malignant cancer:
   clump-t-4: 10, 7, 8, 9.
   u-cell-size-4: 8, 6, 10, 5, 7.
   u-cell-shape-4: 8, 10.
   marginal-adhesion-4: 10, 7.

s-e-cell-size-4: 6, 8, 10.
bare-nuclei-4: 10, 9.
bland-chromatin-4: 8, 7, 9, 6, 10, 5.
normal-nucleoli-4: 10, 8, 9, 6.
mitoses-4: 4, 8.

Based on the discovered concept clusters, the **second stage** is to generate, refine and manage the *classification knowledge* with hierarchical models in a knowledge-base in HML [14]. For example, the hierarchical model of *group* 1 of data that represents the conditions of the benign cancer is shown in Fig.5 and the MLL formula is created to the benign cancer as Rule-1:

Rule-1: /* the rule for diagnosing breast cancer */
[∀ Y#/benign:symptom][∃ X#/Y] p-breast-cancer(Y X).

Rule-1 reads "*if the symptoms recorded in the set-elements relations about benign are satisfied, then the breast cancer is benign*".

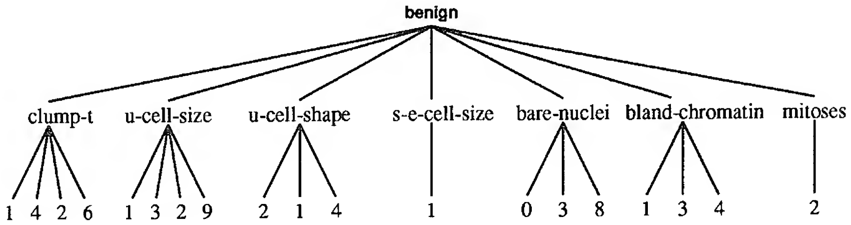


Fig. 5. The hierarchical model of *group* 1

There are two important jobs in knowledge generation. The first is *hierarchical modeling.* Where, the process of representing the concept cluster by the IS-A hierarchy (i.e., the hierarchical model represented by the set-elements relation in MLL, and Fig.5 is an example of its equivalent graph) is called *hierarchical modeling.* For example, two concept clusters discovered from *group* 1 of the breast cancer database can be represented by two IS-A hierarchies. Another is to select quantifiers in the MLL prefix. The theorem, *implication of MLL*, is used for selecting quantifiers ∀ and ∃ [14].

Furthermore, there are two main methods for refining the hierarchical model in HML. First, a better hierarchical model is selected by evaluating the information amount of MLL when new concept clusters are discovered from a database. For example, when another concept cluster as shown in Fig.6 is discovered by adding *group* 2 of data to the breast cancer database, we can calculate the amounts of their information based on the MLL prefix,

$$[\forall\, Y\#/\text{benign:symptom}][\exists\, X\#/Y],$$

for selecting the better one from two hierarchical models shown in Figs. 5 and 6. Since the information amount of the hierarchical model shown in Fig.6 is larger than the one shown in Fig.5, this one shown in Fig.6 is selected. That is, learning is to select a hierarchical model with more information.

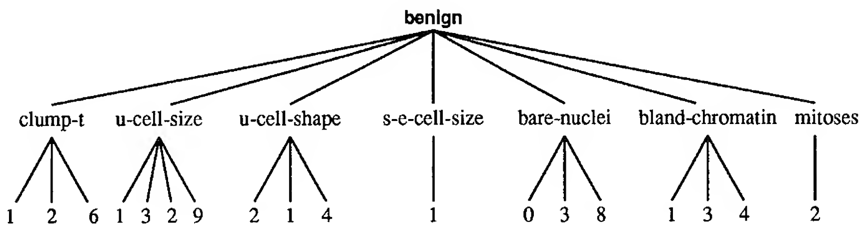


Fig. 6. A hierarchical model in which *group* 2 of data was added

Another method for refinement is that the hierarchical model is refined by cooperative use of domain knowledge and the quantitative evaluation using information theory. For example, if the following domain knowledge,

u-cell-size, u-cell-shape $\subset$ *uniformity*
u-cell-size, u-cell-shape, s-e-cell-size $\subset$ *cell*
bare-nuclei, normal-nucleoli, mitoses $\subset$ *nuclei*
clump-t, bland-chromatin $\subset$ *other*,

is used, then a more refined hierarchical model as shown in Fig.7 can be acquired. That is, the domain knowledge is used for conceptual abstraction (generalization). Furthermore, the prefix of the MLL formula with the hierarchical model as shown in Fig.7 can be represented into

$$[\forall\, Z\#/\text{benign:symptom}][\forall\, Y\#/\text{Z}][\exists\, X\#/Y].$$

Since there are the same quantifiers appeared in succession in the prefix of this formula, the amount of information of this formula with the hierarchical model shown in Fig.7 is the same with the one shown in Fig.6 (according to the theorem, *equivalence of information of MLL*) [14]. Therefore, the hierarchical model shown in Fig.7, in which *group* 2 of data was added and conceptual abstraction was done, is selected as a more refined one. Here, learning is to select the best hierarchical model by using cooperatively domain knowledge and informative evaluation.

Finally, the management of hierarchical models is another important function of HML for managing more hierarchical models generated along with data change (add, delete or update) in databases. In HML, the set chains of hierarchical models and an inheritance graph of hierarchical models are used for this purpose. By means of them, the following jobs can be done: (1) Hierarchical models discovered from databases are first stored in the set chains, and then are

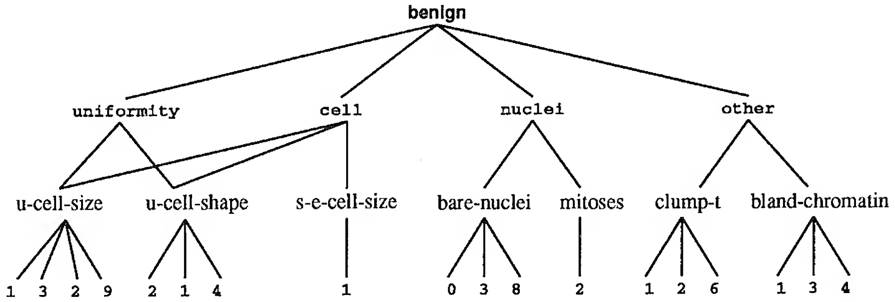


Fig. 7. A hierarchical model used domain knowledge

refined (evaluated/modified) by HML; (2) The time and history of hierarchical models are represented and managed. That is, these set chains for storing hierarchical models are dynamically generated as time goes on for recording the evolution process of hierarchical models; (3) The inheritance graph of hierarchical models is dynamically generated for describing the relationship among hierarchical models. These set chains and the inheritance graph are managed by a meta knowledge level (i.e., *meta-ctl-hml* as shown in Fig.3).

4 Discussion

From the example stated in Sec.3, we can see that the process of discovering *classification knowledge* is essentially the one of information transformation by structuring operations for data in multiple learning phases based on the global learning scheme shown in Fig.2. The result of the transformation is that more generalized form of information, i.e. knowledge, is obtained. In other words, the value of information increases along with the structuring operations in the process of knowledge discovery [5, 8].

However, if we only use DBI and HML in this transformation, there are some limits for discovering knowledge from a very large and complex database. For example, in the real world, there are many real-valued attributes as well as symbolic-valued attributes. When creating the *Probability Distribution Matrix (PDM)* in our method, continuous values must be quantized. On the other hands, in order to discover the better knowledge, conceptual abstraction and generalization are also necessary. Therefore, *attribute oriented clustering* is an useful technique as a step of pre-processing before creating the PDM (i.e., in Phase 1 of the global learning scheme shown in Fig.2) [16]. Another example is to improve the performance of PDM when discovering knowledge from a very large and complex databases. In general, the maximal number of individual data that can be included in a database is $\prod_{i=1}^{m} n_i$ and the size of a PDM is $(\sum_{i=1}^{m} n_i)^2$. Here, m is the number of attributes, n is the number of different data values in each attribute. For example, if we evaluate the breast cancer database shown in Table 1, the maximal number of the database except the attributes *id* and *code-n* is 294912000, the size of the PDM is 5776.

Therefore, we need to use cooperatively more sub-systems (e.g., KOSI and IIBR) and optional parts of GLS for accomplishing a more complex discovering task, organize dynamically the discovery process according to different discovery tasks and improve the performance of GLS itself. Furthermore, we also need to extend GLS into a *multi-agents, distributing* and *cooperating* discovery system for more rational use of computer resources. Thanks to the multiple meta-level structure of GLS as shown in Fig.3 and the development of distributed KAUS (DKAUS) recently [12], GLS as a multi-strategy system can be easily extended into a multi-agents, distributing and cooperating discovery system.

First, the agents in our sense are not static but dynamic, i.e., they are dynamically composed of Intelligent Mail BOX and one optional part, one or partial functions of a sub-system, or some sub-systems of GLS. And their IDs are composed of names of workstation and agent. An agent can try to solve a discovery task or a sub-task, or something that serves this task but necessary to another agent's solving of the task. It can interact with other agents to help solve the task. Second, the communication protocol uses the one of DKAUS that is an extended and revised version of Smith's one [12]. Third, a multiple meta-level structure is used for solving some meta-level problems and controlling the meta or object level. The meta-level problems include mainly decomposing discovery task, allocating resource, adaptive self-configuration of discovering steps, managing interaction/communication among agents, synthesizing part-results of discovery and so on. These meta-level problems are solved by a meta-meta level as shown in Fig.3, i.e., the main roles of the meta-meta level are to organize dynamically a discovery process and control this process as the coordinator.

In comparison, GLS is mostly similar to INLEN in related systems [3]. In INLEN, a database, a knowledge-base and several existing methods of machine learning are integrated as several operators such as data management operators, knowledge management operators and knowledge generation operators. These operators can generate diverse kinds of knowledge about the properties and regularities existing in the data. INLEN was implemented as a toolkit like GLS. However, GLS more stresses the relationships among tools (e.g., DBI and HML, KOSI and IIBR) and can organize dynamically the discovery process according to different discovery tasks. Moreover, the refinement for knowledge is one of important capabilities of GLS that was not developed in current INLEN.

Since the GLS system to be finished by us is very-large, very-complex, we have only finished main parts as *multi-strategy* aspect and have undertaken to extend it into a *multi-agents, distributing* and *cooperating* discovery system. Other future work mainly involves to perfect current sub-systems, develop an intelligent user interface, combine autonomous discovery with interactive discovery, support the process from discovery to invention, and apply our system to more real problems and application fields for further testing and demonstration.

References

1. Matheus, C.J., Chan, P.K., Piatetsky-Shapiro, G. Systems for Knowledge Discovery in Databases. *IEEE Trans. Knowl. Data Eng.*, Vol.5 (No.6), (1993) 904-913.
2. Mangasarian, O.L., Wolberg, W.H. Cancer diagnosis via linear programming. *SIAM news*, Vol.23(No.5), (1990) 1-18.
3. Michalski, R.S., Kerschberg, L., Kaufman, K.A., Ribeiro, J.S. Mining for Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results. *J. of Intell. Infor. Sys.*, Vol.1(No. 1), (Kluwer Academic Publishers, 1992) 85-113.
4. Michalski, R.S., Carbonell, J.G., Mitchell, T.M. et al. *Machine Learning - An Artificial Intelligence Approach*, Vols.1,2,3,4. (1983/86/90/94).
5. Ohsuga, S. A Consideration to Knowledge Representation - An Information Theoretic View. *Bulletin of Informatics and Cybernetics*, Vol.21(No.1-2), (1984) 121-135.
6. Ohsuga, S., Yamauchi, H. Multi-Layer Logic - A Predicate Logic Including Data Structure as Knowledge Representation Language. *New Generation Computing*, Vol.3(No.4), (1985) 403-439.
7. Ohsuga, S. Framework of Knowledge Based Systems. *Knowl. Based Sys.*, Vol.3(No.4), (1990) 204-214.
8. Ohsuga, S. From Data to Knowledge - A Formal Approach for Knowledge Discovery. *Proc. Symposium of Advanced in Knowl. Sci.*, (1993) 47-53.
9. Piatetsky-Shapiro, G., Frawley, W.J. (eds.). *Knowledge Discovery in Databases.* AAAI/MIT Press, (1991).
10. Piatetsky-Shapiro, G., Matheus, C.J. Knowledge Discovery Workbench for Exploring Business Databases. *Int. J. of Intell. Sys.*, Vol.7(No.7), (1992) 675-686.
11. Shavlik, J.W., Dietterich, T.G. (eds.). *Readings in Machine Learning.* MORGAN KAUFMANN PUBLISHERS, INC., (1990).
12. Suzuki, E., Hori, K., Ohsuga, S., Morizet-Mahoudeaux, P. Problem Solving by Negotiation among Autonomous Knowledge Processing Systems. *J. of Japanese Society for Artif. Intell.*, Vol.9 (No.1), (1994) 109-118.
13. Zhong, N., Ohsuga, S. GLS - A Methodology for Discovering Knowledge from Databases. *Proc. 13th Int. CODATA Conf. entitled "New Data Challenges in Our Information Age"*, (1992).
14. Zhong, N., Ohsuga, S. HML - An Approach for Managing/Refining Knowledge Discovered from Databases. *Proc. 5th IEEE Int. Conf. on Tools with Artif. Intell. (TAI'93)*, (1993) 418-426.
15. Zhong, N., Ohsuga, S. An Integrated Calculation Model for Discovering Functional Relations from Databases. *V. Marik, et al. (eds.) Database and Expert Systems Applications. Proc. 4th Int. Conf., DEXA'93.* Lecture Notes in Computer Science 720, (Springer-Verlag, 1993) 213-220.
16. Zhong, N., Ohsuga, S. Discovering Concept Clusters by Decomposing Databases. *Data & Knowl. Eng.*, Vol.12(No.2), (North-Holland, 1994) 223-244.
17. Zhong, N., Ohsuga, S. KOSI - An Integrated Discovery System for Discovering Functional Relations from Databases. *manuscript.*
18. Zhong, N., Ohsuga, S. IIBR - A System for Managing/Refining Functional Relations Discovered from Databases. *manuscript.*
19. Zytkow, J.M., Zembowicz, R. Database Exploration in Search of Regularities. *J. of Intell. Infor. Sys.*, Vol.2(No. 1), (Kluwer Academic Publishers, 1993) 39-81.
20. Zytkow, J.M. Introduction: Cognitive Autonomy in Machine Discovery. *Machine Learning*, Vol.12(Nos.1/2/3), (Kluwer Academic Publishers, 1993) 7-16.

Declarative Semantics for Contradictory Modular Logic Programs

Anastasia Analyti & Sakti Pramanik
Department of Computer Science, Michigan State University, E. Lansing, MI 48824
E-mail: analyti@cps.msu.edu, pramanik@cpswh.msu.edu

Abstract. A complex reasoning system can be designed as an interaction between reasoning modules. A module consists of a declaration of exported/imported predicates and a set of rules containing both negation by default and classical negation. A *prioritized modular logic program* (*PMP*) consists of a set of modules and a partial order $<_{\text{def}}$ on the predicate definitions (M, p), where M is a module and p is a predicate exported by M. Because of the classical negation, conflicts may arise within and among modules. The partial order $<_{\text{def}}$ denotes the relative reliability of the predicate definitions contributing to the conflict. We present the *reliable semantics* for *PMPs*. The goal of the reliable semantics is to draw reliable conclusions from possibly contradictory *PMPs*.

1 Introduction

Modules in a reasoning system arise as a result of a functional decomposition of a complex reasoning task into a number of simpler subtasks. Each module is an interactive reasoning subsystem that is used for the (often partial) *definition* of its exported predicates. Each module contains a set of rules viewed as an open logic theory [1] with a set of input literals. A module represents an incomplete specification of some domain because its input literals are defined in other modules. However, a module becomes a standard extended logic program (closed module) when the truth values of its input literals are known.

A *prioritized modular logic program* (*PMP*) consists of a set of modules and a partial order $<_{\text{def}}$ on the predicate definitions (M, p), where M is a module and p is a predicate exported by M. We assume that modules are internally consistent. However, a *PMP* is possibly not globally consistent. When a conflict occurs, $<_{\text{def}}$ expresses our relative confidence in the predicate definitions contributing to the conflict. Each module has a set of *local* literals that are inaccessible to other modules. Literals that can be accessed (imported) by any module have the form $\{M_1,...,M_n\}$:A, $\{M_1,...,M_n\}$: $\neg A$ or $\{M_1,...,M_n\}$: $\sim A$, where M_i are module names and A is a conventional atom whose predicate is exported by all M_i. Intuitively, a literal $\{M_1,...,M_n\}$:A is evaluated as true if (i) A is derived from a module M_i, and (ii) if $\neg A$ is derived from a module $M_j \neq M_i$ then result A has higher priority than result $\neg A$. A literal $\{M_1,...,M_n\}$: $\sim A$ is evaluated as true if $\{M_1,...,M_n\}$: $\neg A$ is true or $\sim A$ is true in all modules M_i.

We present a semantics for *PMPs*, called *reliable semantics* (*RS*), which assigns a truth value *true*, *false* or *unknown* to every literal. Every *PMP* has at least one *stable m-model*. The *RS* of a program P is the least fixpoint of a monotonic operator and the least (w.r.t. $\subseteq$) stable m-model of P. When a *PMP* is contradictory, exported results (represented by indexed literals) are considered *unreliable* if: (i) they contribute to a contradiction, and (ii) they do not have higher priority than the other contributing results. The *RS* of a *PMP*, *P*, represents the skeptical meaning of *P* and thus, none of its conclusions is based on unreliable exported results. Credulous conclusions are obtained by isolating the conflicting results in the multiple stable *m*-models of *P*.

2 Informal Presentation and Intuitions

Our framework can be used for the representation of result-sharing cooperating agents [MaJo89]. A complex task is statically decomposed into a set of simpler subtasks, each assigned to an agent. Agents may have overlapping or even identical capabilities. Therefore, it is possible that they export agreeing or contradictory results. When agents M_1, M_2 export contradictory conclusions about a literal L, the truth value of L w.r.t. M_1, M_2 (expressed by the truth value of the literal $\{M_1, M_2\}{:}L$) is unknown. Yet, agents M_1, M_2 maintain their individual beliefs about L which is expressed by the truth value of the literals $\{M_1\}{:}L$, $\{M_2\}{:}L$, respectively.

Example 2.1: Sensors S_1, S_2 are gathering information from two different angles about the persons entering a building. Modules M_1, M_2 are assigned with the identification of terrorists based on the information collected from sensors S_1, S_2, respectively. Each module M_1, M_2 exports the result *entered*(*terrorist*) (resp. ¬*entered*(*terrorist*)) iff it reaches the conclusion that an (resp. no) terrorist has entered the building. It is possible that M_1, M_2 disagree, i.e., M_1 exports *entered*(*terrorist*) whereas M_2 exports ¬*entered*(*terrorist*). The results exported by M_1, M_2 can be queried by other modules in various ways. For example,

- $Query_1 \leftarrow \{M_1\}{:}entered(terrorist), \{M_2\}{:}entered(terrorist).$

  $Query_1$ is true if *entered*(*terrorist*) is true in both M_1, M_2. $Query_1$ is false by default if *entered*(*terrorist*) is false (by default or classically) in at least one of M_1, M_2.

- $Query_2 \leftarrow \{M_1, M_2\}{:}\ entered(terrorist).$

  $Query_2$ is true if *entered*(*terrorist*) is true in at least one of M_1, M_2 and M_1, M_2 do not disagree. $Query_2$ is false by default if *entered*(*terrorist*) is false in both M_1, M_2.

- $Query_3 \leftarrow \{M_1\}{:}\ \sim entered(terrorist).$

  $Query_3$ is true if *entered*(*terrorist*) is false in M_1 (even if *entered*(*terrorist*) is true in M_2).

Individual agent theories are assumed to be consistent. Yet, the consistency of the union of agent theories is not assured. As we saw in Example 2.1, one case of contradiction is when independent modules export contradictory results. In this case, the contradiction depends only on the independent modules and it is relatively easy to resolve. Yet, generally, contradictions may involve several module interactions. For example, an agent exports a faulty result, this result is imported by another agent which exports a faulty result based on the imported faulty result. After a few module interactions, contradiction may arise in two ways : (i) Complementary literals are derived inside an module. (ii) Complementary literals are exported from two different modules.

When contradiction appears, the sources of the contradiction are traced back to the contributing exported results. Domain specific information might indicate that some exported results are more reliable (have higher priority) than others. Let res_1 and

res_2 be two exported results contributing to the contradiction. If res_1 has higher priority than res_2 and no contradiction arises without res_2 then only res_1 is taken into account. If the priority of res_1, res_2 cannot be compared then both are eliminated from RS (skeptical approach).

3. *m*-models for Prioritized Modular Logic programs

Our alphabet contains a finite set of constant, predicate and variable symbols from which ordinary atoms are constructed in the usual way. It also contains a finite set of module names. An *indexed atom* has the form $\{M_1,...,M_n\}:A$, where A is an ordinary atom and M_i are module names. A *classical literal* is either an atom A or its classical negation $\neg A$. The classical negation of a literal L is denoted by $\neg L$, where $\neg(\neg L)=L$. A *default literal* is the default negation $\sim L$ of a classical literal L, where $\sim(\sim L)=L$. A literal is called *indexed* when its corresponding atom is indexed. We define $\{M_1,...,M_n\}:\neg A = \neg\{M_1,...,M_n\}:A$ and $\{M_1,...,M_n\}:\sim A= \sim\{M_1,...,M_n\}:A$, where M_i are module names and A is an ordinary atom. The classical literals L, $\neg L$ are called *complementary* to each other. The predicate of a literal L is denoted by $pred_L$.

A module with name M is a tuple $<Exp_M, Imp_M, R_M>$. The set Exp_M contains the predicates that are exported (defined) by M. The set Imp_M contains the predicates imported by M. The set R_M contains rules of the form: $L\leftarrow L_1,...,L_m,\sim L_{m+1},...,\sim L_n$, where L is a non-indexed classical literal and L_i are classical literals. If an indexed literal $\{M_1,...,M_n\}:L$ is in the body of a rule then M imports L from the modules $M_1,...,M_n$. If a non-indexed literal L is in the body of a rule and $pred_L\in Imp_M$ then M imports L from all modules that export $pred_L$.

A *prioritized modular logic program* (*PMP*), P, is a pair $<Mod_P, <_{\text{def}}>$. Mod_P is a set of modules and $<_{\text{def}}$ a partial order on Def_P, where $Def_P=\{(M,p) \mid M\in Mod_P$ and $p\in Exp_M\}$. Each pair $(M,p)\in Def_P$ represents the *definition* of predicate p in module M. If $\{M_1,...,M_n\}:L$ is an indexed literal appearing in P then $(M_i,pred_L)\in Def_P$, $\forall i\leq n$.

Individual modules are assumed to be consistent but their union may be inconsistent. Thus, when complementary literals are derived within a module M, it is because of unreliable information imported by M. When literals L, $\neg L$ are derived from different modules M, M', it is because the definition of $pred_L$ in M, M' is unreliable or the information imported by M, M' is unreliable. When conflict occurs, $<_{\text{def}}$ expresses our relative confidence in the predicate definitions contributing to the conflict. Let (M,p) and (M',p') be in Def_P. The notation $(M,p) < (M',p')$ (resp. $(M,p) \not< (M',p')$) means that the definition of p in M is less (resp. not less) trusted than that of p' in M'. Intuitively, a literal L exported by a module M is reliable if it does not contribute to any derivation of complementary literals caused by definitions $(M',p') \not< (M,pred_L)$. Note that, the reliability of L is not affected by predicate definitions less trusted than the definition of $pred_L$ in M.

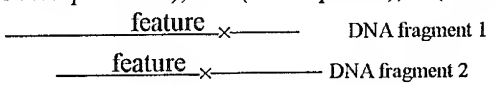
We define $S_L=\{M\in Mod_P \mid pred_L\in Exp_M\}$, where L is a literal. The indexed literals $S_L:L$ are called *globally indexed*. To simplify the presentation of the semantics a *renaming* mechanism is employed. Let r be a rule in a module M. Then, the head L

of *r* is replaced by a new literal $M\#L$. Every non-indexed literal L in the body of *r* with $pred_L \notin Imp_M$ is replaced by $M\#L$. Every non-indexed literal L in the body of *r* with $pred_L \in Imp_M$ is replaced by the globally indexed literal $S_L{:}L$. Literals $M\#L$ are called *local* to M and they are not accessed by other modules. In contrast to local literals, *indexed* literals are accessible to all modules. When we refer to a *PMP*, we assume that the above renaming has already been done. Note that after renaming, only local and indexed literals appear in a module M. We define $M\#\neg A = \neg M\#A$ and $M\#{\sim}A = {\sim}M\#A$, where M is a module name and A is an ordinary atom.

The set of all instantiations of the classical literals appearing in P and the globally indexed classical literals $S_L{:}L$, where L appears in P, is called the *Herbrand Base* (HB_P) of P. Let M be a module. We define $close_u(M)$ as the extended program that results if we replace every indexed literal in M with the special proposition u (meaning *undefined*). We say that M is ***internally consistent*** iff the *extended well-founded semantics* [PeAl92] of $close_u(M)$ is defined (the *WFM* of $close_u(M)$ is not contradictory). All modules of a *PMP* are assumed to be internally consistent.

If S is a set of literals then ${\sim}S = \{{\sim}L \mid L \in S\}$ and $\neg S = \{\neg L \mid L \in S\}$. The notations $Head_r$ and $Body_r$ denote the head and body of a rule *r*.

Example 3.1: A *contig* is a set of overlapping DNA fragments that span some region of a genome. One method to detect overlaps uses common features. The idea is that if the same feature is found in two DNA fragments then they probably overlap. The overlap is not certain because of unreliable experimental results and feature repetition in the genome. Consider the following *PMP* (before renaming) with modules *OF* (*OverlapFeature*), *OD* (*OverlapData*), *F* (*Feature*): (variables start with capital)



module *OF* **exports** *overlap* **imports** *feat*
/* If the same feature *Feat* is found in *Frag*1 and *Frag*2 then the fragments overlap */
rules $overlap(Frag1,Frag2) \leftarrow feat(Frag1,Feat), feat(Frag2,Feat)$.

module *OD* **exports** *overlap*
/* Very reliable *Overlap* data */
rules $\neg overlap(frag1,frag2)$.

module *F* **exports** *feat*
/* A feature *x* is found in *frag*1 and *frag*2 */
rules $feat(frag1, x)$. $feat(frag2, x)$.

$(F,feat) < (OD,overlap)$ /* *overlap* data in *OD* are more reliable than data in *F* */

Even though the modules *OD*, *OF* and *F* are internally consistent, their union is inconsistent because both $overlap(frag1,frag2)$ and $\neg overlap(frag1,frag2)$ can be derived from the above rules. Note that $\neg overlap(frag1,frag2)$ is derived from module *OD* and that the derivation of $overlap(frag1,frag2)$ from module *OF* depends on $feat(frag1,x)$ and $feat(frag2,x)$, exported by module *F*. Since $(F,feat) < (OD,overlap)$, i.e., the definition of *feat* in *F* is less reliable that of *overlap* in *OD*, the results $feat(frag1,x)$ and $feat(frag2,x)$ are considered unreliable. Thus, the truth value of the literals $feat(frag1,x)$ and $feat(frag2,x)$ is considered unknown and the literal $\neg overlap(frag1,frag2)$ is evaluated as true. After the renaming mechanism is employed, modules become:

module *OF* **exports** *overlap* **imports** *feat*
rules $OF\#overlap(Frag1,Frag2) \leftarrow \{F\}{:}feat(Frag1, Feat), \{F\}{:}feat(Frag2, Feat)$.

module *OD* **exports** *overlap*
rules $\neg OD\#overlap(frag1,frag2)$.

module *F* **exports** *feat*
rules $F\#feat(frag1, x)$. $F\#feat(frag2, x)$.

Definition 3.1 (interpretation): Let P be a *PMP*. An interpretation I is a set of literals $T \cup \sim F$, where T and F are disjoint sets of classical literals. I is *consistent* iff $T \cap \neg T = \emptyset$. I is *coherent* iff it satisfies the *coherence property*: $\neg T \subseteq F$.

In the above definition, T contains the *classically true* literals, $\neg T$ contains the *classically false* literals and the set F contains the literals *false by default*. The *coherence operator* (*coh*) [PeAl92] is used to transform an interpretation to a coherent one. If I is a set of literals then $coh(I) = I \cup \{\sim L \mid \neg L \in I\}$.

Definition 3.2 (truth valuation of a literal): A literal L is true (resp. false) w.r.t. an interpretation I iff $L \in I$ (resp. $\sim L \in I$). A literal that is neither true nor false w.r.t. I, it is undefined w.r.t. I.

An interpretation I can be seen equivalently as a function from the set of classical literals to $\{0,1/2,1\}$, where $I(L)=1$ when L is true w.r.t. I, $I(L)=0$ when L is false w.r.t. I and $I(L)=1/2$ when L is undefined w.r.t. I. We define $I(\emptyset)=1$ and $I(S)= \min\{I(L) \mid L \in S\}$, where S is a non-empty set of literals.

The truth value of a literal $M\#L$ represents the truth value of L in module M. A classical literal $\{M_1,\dots,M_n\}{:}L$ is true iff $M_i\#L$ is true for an $i \leq n$ and $\{M_1,\dots,M_n\}{:}L$ is reliable. A default literal $\sim\{M_1,\dots,M_n\}{:}L$ is true iff $\sim M_i\#L$ is true for all $i \leq n$ and $\sim\{M_1,\dots,M_n\}{:}L$ is reliable. Let I be a set of literals known to be true. In Def. 3.5, the concept of reliable indexed literal is defined which is used in defining the m-models and in the fixpoint computation of the reliable semantics. To decide if an indexed literal $S{:}L$ is reliable w.r.t. a definition (M,p), all possible derivations of complementary literals caused by definitions $(M',p') \not< (M,p)$ should be considered. $S{:}L$ is reliable if it does not contribute to any such derivation of complementary literals. Intuitively, $\mathrm{Pos}_{[M,p],I}$ contains the literals possible to derive when results exported by modules M' with $(M',pred_L) < (M,p)$ are ignored.

Definition 3.3 (possible literal set): Let P be a *PMP*, I,J sets of literals and $(M,p) \in Def_P$. The *possible literal set* w.r.t. (M,p) and I, denoted by $\mathrm{Pos}_{[M,p],I}$, is defined as follows:

- $\mathbf{PT}_{[M,p],I}(J) = \{M'\#L \mid \exists\, M'\#L \leftarrow L_1,\dots,L_n$ in module M' s.t. $L_i \in J,\ \forall i \leq n\} \cup$
  $\{S{:}L \in HB_P \mid \neg S{:}L \notin I$ and $\exists\, M' \in S$ s.t. $M'\#L \in J$ and $(M',pred_L) \not< (M,p)\}$
- $\mathbf{PF}_{[M,p],I}(J)$ is the greatest set $U \subseteq HB_P$ s.t.
  (i) if $M'\#L \in U$ and r is a rule s.t. $Head_r = M'\#L$ then $\exists K \in Body_r$ s.t. $K \in U$ or $\sim K \in J$,
  (ii) if $S{:}L \in U$ then $\forall M' \in S$, $M'\#L \in U$ and $(M',pred_L) \not< (M,p)$.
- $\mathbf{PW}_{[M,p],I}(J) = coh(\mathbf{PT}_{[M,p],I}(J) \cup \sim\mathbf{PF}_{[M,p],I}(J))$.
- $\mathrm{Pos}_{[M,p],I}$ is the least (w.r.t. $\subseteq$) fixpoint of the operator $\mathbf{PW}_{[M,p],I}$.

Intuitively, a literal $S{:}L$ is reliable w.r.t. (M,p) and I iff there are no K, $\neg K$ in $\mathrm{Pos}_{[M,p],I}$ s.t. the derivation of K depends on $S{:}L$. The *dependency set* of K w.r.t. (M,p)

and I is the set of literals in $\mathrm{Pos}_{[M,p],I}$ that the derivation of K depends on. Since I is a set of literals known to be true, the *dependency set* of a literal $K \in I$ equals $\{\}$.

Definition 3.4 (dependency set): Let P be a *PMP*, I a set of literals and $(M,p) \in Def_P$. The *dependency set* of a literal K w.r.t. (M,p) and I, denoted by $\mathrm{Dep}_{[M,p],I}(K)$, is the least (w.r.t. $\subseteq$) set $\mathrm{D}(K)$ s.t. if $K \in I$ then $\mathrm{D}(K)=\{\}$ else

(i) If $K= {\sim}M'\#L$ is a default literal then

(a) $\mathrm{D}(\neg M'\#L) \subseteq \mathrm{D}({\sim}M'\#L)$ and

(b) $\forall\, M'\#L \leftarrow L_1,\ldots,L_n$ in M', if ${\sim}L_i \in \mathrm{Pos}_{[M,p],I}$ for $i \leq n$ then $\mathrm{D}({\sim}L_i) \subseteq \mathrm{D}({\sim}M'\#L)$.

(ii) If $K=M'\#L$ is a classical literal then

if $M'\#L \leftarrow L_1,\ldots,L_n$ in M' s.t. $\{L_1,\ldots,L_n\} \subseteq \mathrm{Pos}_{[M,p],I}$ then $\mathrm{D}(L_i) \subseteq \mathrm{D}(M'\#L)$, $\forall i \leq n$.

(iii) If $K= {\sim}S{:}L$ is a default literal then

(a) $\mathrm{D}(\neg S{:}L) \subseteq \mathrm{D}({\sim}S{:}L)$ and

(b) if $\forall M' \in S$, $(M',pred_L) \not< (M,p)$ and ${\sim}M'\#L \in \mathrm{Pos}_{[M,p],I}$ then

$\mathrm{D}({\sim}M'\#L) \subseteq \mathrm{D}({\sim}S{:}L)$, $\forall M' \in S$ and ${\sim}S'{:}L \in \mathrm{D}({\sim}S{:}L)$, $\forall S' \subseteq S$.

(iv) If $K= S{:}L$ is a classical literal then

(a) $\mathrm{D}(M'\#L) \subseteq \mathrm{D}(S{:}L)$, $\forall M' \in S$ s.t. $(M',pred_L) \not< (M,p)$ and $S'{:}L \in \mathrm{D}(S{:}L)$, $\forall S' \subseteq S$.

Definition 3.5 (reliable indexed literal): Let P be a *PMP*, I a set of literals, $S{:}L$ a literal, $(M,pred_L) \in Def_P$, $M \in S$ and p be equal to $pred_L$.

- The literal $S{:}L$ is *unreliable* w.r.t. (M,p) and I iff $\exists\, \neg K \in \mathrm{Pos}_{[M,p],I}$ s.t.
  if $K = S'{:}L$ with $S' \supset S$ then $S{:}L \in \mathrm{Dep}_{[M,p],I}(M'\#L)$ for an $M' \in S'$ s.t. $(M',p) \not< (M,p)$
  else $S{:}L \in \mathrm{Dep}_{[M,p],I}(K)$.
- The literal $S{:}L$ is *reliable* w.r.t. (M,p) and I iff it is not *unreliable* w.r.t. (M,p) and I.

Assume that $S{:}L$ is *unreliable* w.r.t. (M,p) and I. If K of Definition 3.5 is a local literal $M'\#L'$ then (i) the literals K, $\neg K$ are derived inside module M' and (ii) $S{:}L$ contributes to the derivation of K. If $K = S'{:}L' \neq S{:}L$ is an indexed literal then: (i) there are literals $M'\#L'$, $\neg M''\#L'$ derived in modules $M' \in S'$ and $M'' \in S'$ s.t. $(M',p) \not< (M,p)$ and $(M'',p) \not< (M,p)$, and (ii) $S{:}L$ contributes to the derivation of $M'\#L'$. If $K = S{:}L$ then there are literals $M'\#L$, $\neg M''\#L$ derived in modules $M' \in S$ and $M'' \in S$ with $(M',p) \not< (M,p)$ and $(M'',p) \not< (M,p)$.

Example 3.2: Let P be the *PMP* of Example 3.1 and $I=\emptyset$.

$\mathrm{Pos}_{[OD,overlap],I} = coh(\{\neg OD\#overlap(frag1,frag2),\ \ \neg\{OD\}{:}overlap(frag1,frag2),$
$\neg\{OD,OF\}{:}overlap(frag1,frag2)\})$

The literal $\neg\{OD\}{:}overlap(frag1,frag2)$ is reliable w.r.t. $(OD,overlap)$ and I because
$\neg\{OD\}{:}overlap(frag1,frag2) \notin \mathrm{Dep}_{[OD,overlap],I}(\neg K)$, $\forall K \in \mathrm{Pos}_{[OD,overlap],I}$.

Similarly, $\neg\{OD,OF\}{:}overlap(frag1,frag2)$ is reliable w.r.t. $(OD,overlap)$ and I.

$\mathrm{Pos}_{[F,feat],I} = coh(\{F\#feat(frag1,x),\ \ \{F\}{:}feat(frag1,x),\ \ F\#feat(frag2,x),$
$\{F\}{:}feat(frag2,x),\ \ OF\#overlap(frag1,frag2),\ \ \{OF\}{:}overlap(frag1,frag2),$
$\{OD,OF\}{:}overlap(frag1,frag2),\ \ \neg OD\#overlap(frag1,frag2),$
$\neg\{OD\}{:}overlap(frag1,frag2),\ \ \neg\{OD,OF\}{:}overlap(frag1,frag2)\})$

The literal $\{F\}{:}feat(frag1,x)$ is unreliable w.r.t. $(F,feat)$ and I because

(i) $\neg\{OD,OF\}{:}overlap(frag1,frag2) \in \mathrm{Pos}_{[F,feat],I}$ and

(ii) $\{F\}{:}feat(frag1,x) \in \mathrm{Dep}_{[F,feat],I}(\{OD,OF\}{:}overlap(frag1,frag2))$.

Similarly, $\{F\}{:}feat(frag2,x)$ is unreliable w.r.t. $(F,feat)$ and I.

$\mathrm{Pos}_{[OF,overlap],I} = \mathrm{Pos}_{[F,feat],I}$. The literal $\{OF\}{:}overlap(frag1,frag2)$ is reliable w.r.t. $(OF,overlap)$ and I whereas the literal $\{OD,OF\}{:}overlap(frag1,frag2)$ is not.

Definition 3.6 (*m*-model): Let P be a *PMP*. A consistent, coherent interpretation I is an *m-model* of P iff (i) $\forall$ rule r, $I(Head_r) \geq I(Body_r)$ and (ii) $\forall$ classical literal $S{:}L$, both of the following are true:

- if $I(S{:}L)\neq 1$ then $\forall M\in S$ s.t. $I(M\#L)=1$, $S{:}L$ is unreliable w.r.t. $(M,pred_L)$ and I
- if $I(S{:}L)=0$ then $I(\neg S{:}L)=1$ or $\forall M\in S$, $I(M\#L)=0$.

Since condition (i) defines 3-valued models [9], an *m*-model of P is a 3-valued model of every module of P. In condition (ii), the first subcondition expresses that if $S{:}L$ is a classical literal, $M\in S$ and $I(M\#L)=1$ then $I(S{:}L)$ can be $\neq 1$ only if $S{:}L$ is unreliable w.r.t. $(M,pred_L)$ and I. The purpose of the second subcondition in condition (ii) is to allow $S{:}L$ to be false when $\neg S{:}L$ holds, even if $\exists M\in S$ s.t. $I(M\#L)>0$.

Example 3.3: Let P be as in Example 3.1. Then, I is an *m*-model of P where $I=coh(\{F\#feat(frag1,x),\ F\#feat(frag2,x),\ \neg OD\#overlap(frag1,frag2),\ \neg\{OD\}{:}overlap(frag1,frag2),\ \neg\{OD,OF\}{:}overlap(frag1,frag2)\})$.

4. Reliable Semantics for Prioritized Modular Logic Programs

In this Section, we define the *reliable model*, *stable m-models* and *reliable semantics* of a *PMP*, P. We show that the reliable model of P is the least (w.r.t. $\subseteq$) stable *m*-model of P.

Definition 4.1 (*m*-unfounded set): Let P be a *PMP* and J a set of literals. A literal set $U \subseteq HB_P$ is *m-unfounded* w.r.t. J iff

(i) if $M\#L\in U$ and r is a rule with $Head_r=M\#L$ then $\exists K\in Body_r$ s.t. $K\in U$ or $\sim K\in J$,

(ii) if $S{:}L\in U$ then $\forall M\in S$, $M\#L\in U$ and $\sim S{:}L$ is reliable w.r.t. $(M,pred_L)$ and J.

The $\mathbf{W}_P$ operator extends the $\mathbf{W}_P$ operator of the *WFS* [11], to *PMPs*.

Definition 4.2 ($\mathbf{W}_P$ operator): Let P be a *PMP* and J a set of literals. We define:

- $\mathbf{T}(J)=\{M\#L \mid \exists\, M\#L\leftarrow L_1,\ldots,L_n$ in module M s.t. $L_i\in J,\ \forall i\leq n\} \cup$
  $\{S{:}L\in HP_P \mid M\#L\in J,\ M\in S$ and $S{:}L$ is reliable w.r.t. $(M,pred_L)$ and $J\}$
- $\mathbf{F}(J)$ is the greatest *m*-unfounded set w.r.t. J.
- $\mathbf{W}_P(J)=coh(\mathbf{T}(J)\cup\sim\mathbf{F}(J))$.

The union of two *m*-unfounded sets w.r.t. J is an *m*-unfounded set w.r.t. J. So, $\mathbf{F}(J)$ is the union of all *m*-unfounded sets w.r.t. J. We define the transfinite sequence $\{I_a\}$ as follows: $I_0=\{\}$, $I_{a+1}=\mathbf{W}_P(I_a)$ and $I_a=\cup\{I_b \mid b<a\}$ if a is a limit ordinal.

Proposition 4.1: Let P be a *PMP*. $\{I_a\}$ is a monotonically increasing (w.r.t. $\subseteq$) sequence of consistent, coherent interpretations of P.

Since $\{I_a\}$ is monotonically increasing, there is a smallest ordinal d s.t. $I_d=I_{d+1}$.

Proposition 4.2: Let P be a *PMP*. Then, I_d is an *m*-model of P.

Definition 4.3 (reliable semantics): Let P be a *PMP*. The *reliable model* of P, RM_P, is the m-model I_d. The *reliable semantics* of P is the "meaning" represented by RM_P.

It is possible that a local literal $M\#L \in RM_P$ but $\{M\}{:}L \notin RM_P$. This, intuitively, means that module M concludes L but that conclusion may be erroneous.

Example 4.1: Let P be the program of Example 3.1 and I be the m-model of Example 3.3. Then, I is the reliable model of P. When $<_{\mathrm{def}}=\{\}$, $RM_P = coh(\{F\#feat(frag1,x), F\#feat(frag2,x), \neg OD\#overlap(frag1,frag2)\})$.

Proposition 4.3: Let P be a *PMP*. The complexity of RM_P is polynomial w.r.t. $|P|$.

The reliable model of a *PMP* corresponds to its skeptical meaning. Credulous meanings can be obtained using the transformation $P/_m I$, where I is an interpretation of P. The transformation P/I is defined in [5, 9] for a normal program P.

Definition 4.4 (transformation $P/_m I$): Let P be a *PMP* and I an interpretation. The program $P/_m I$ is obtained from P as follows:

(i) Remove all rules that contain in their body a default literal $\sim L$ s.t. $I(L)=1$.

(ii) Remove any rule r s.t. $I(\neg Head_r)=1$.

(iii) Remove from the body of the remaining rules any default literal $\sim L$ s.t. $I(L)=0$.

(iv) Replace all remaining default literals $\sim L$ with u.

(v) For all $S{:}L \in HB_P$ s.t. $I(S{:}L)=1/2$,

- for all $M \in S$, if $I(M\#L)=1$ then add $S{:}L \leftarrow u$ else add $S{:}L \leftarrow M\#L$,
- if $\exists M \in S$ s.t. $\sim S{:}L$ is unreliable w.r.t. $(M, pred_L)$ and I then add $S{:}L \leftarrow u$.

(vi) For all $S{:}L \in HB_P$ s.t. $I(S{:}L) \neq 1/2$ and $I(\neg S{:}L) \neq 1$, add $S{:}L \leftarrow M\#L$, $\forall M \in S$.

We say that a stable m-model I of P is the *least*$_v$ *model* of P iff $I(L) \leq I'(L)$, for any stable m-model I' and classical literal L of P.

Definition 4.5 (stable m-model): Let P be a *PMP* and I an m-model of P. I is a *stable m-model* of P iff $least_v(P/_m I)=I$.

Let I be a stable m-model of P. If $S{:}L$ is unreliable w.r.t. $(M, pred_L)$ and I, for an $M \in S$ then the truth value of $S{:}L$ can be unknown w.r.t. I even if $I(M\#L)=1$. If $\sim S{:}L$ is unreliable w.r.t. $(M, pred_L)$ and I, for an $M \in S$ then the truth value of $S{:}L$ can be unknown w.r.t. I even if $I(M\#L)=0$, for all $M \in S$.

The *export rule set* of P is defined as $ER_P = \{S{:}L \leftarrow M\#L \mid S{:}L \in HB_P \text{ and } M \in S\} \cup \{\sim S{:}L \leftarrow \sim M_1\#L, \ldots, \sim M_n\#L \mid S{:}L \in HB_P \text{ and } S=\{M_1, \ldots, M_n\}\}$. An interpretation I of P satisfies $r \in ER_P$ iff $I(Body_r) \neq 1$ or $I(Head_r)=1$. Let I_1, I_2 be two stable m-models of P. We say that $I_1 \leq_{\mathrm{sat}} I_2$ iff (i) $\forall r \in ER_P$, if I_1 satisfies r then I_2 satisfies r and (ii) $I_1 \subseteq I_2$ or $\exists r \in ER_P$ s.t. I_2 satisfies r and I_1 does not satisfy r. In other words, $I_1 \leq_{\mathrm{sat}} I_2$ iff I_2 satisfies more export rules than I_1 or ($I_1 \subseteq I_2$ and I_2 satisfies the same export rules as I_1). Maximal (w.r.t. $\leq_{\mathrm{sat}}$) stable m-models can be seen as the credulous meanings of P.

Example 4.2: Let P be the program of Example 3.1. Then P has four stable m-models:
$I_1=RM_P$, $\quad I_2=RM_P \cup coh(\{\{F\}{:}feat(frag1,x)\})$,
$I_3=RM_P \cup coh(\{\{F\}{:}feat(frag2,x)\})$ and $I_4=RM_P \cup coh(\{\{F\}{:}feat(frag1,x), \{F\}{:}feat(frag2,x), OF\#overlap(frag1,frag2), \{OF\}{:}overlap(frag1,frag2)\})$.

I_2, I_3 and I_4 are maximal (w.r.t. $\leq_{sat}$) stable *m*-models of *P*. Note that
Model I_2 does not satisfy $\{F\}$:*feat*(*frag*2,*x*)$\leftarrow$ *F*#*feat*(*frag*2,*x*),
Model I_3 does not satisfy $\{F\}$:*feat*(*frag*1,*x*)$\leftarrow$ *F*#*feat*(*frag*1,*x*) and
Model I_4 does not satisfy $\{OD,OF\}$:*overlap*(*frag*1,*frag*2)$\leftarrow$ *OF*#*overlap*(*frag*1,*frag*2).

Proposition 4.4: Let *P* be a *PMP*. The reliable model of *P* is a stable *m*-model of *P*.

Proposition 4.5: The reliable model of a *PMP* is its least (w.r.t. $\subseteq$) stable *m*-model.

According to *RS* presented in the previous sections, the confidence in a globally indexed default literal $\sim L$, derived by the default rule for negation, depends on the minimal priorities of $(M,\ pred_L) \in Def_P$. Thus, in case of conflict, $\sim L$ may not be considered less reliable than literals that their derivation is not based on closed-world assumptions. When this is undesirable for a set of predicates $Pred_\sim$, a new module $M_\sim$ can be added which has no rules but exports all predicates in $Pred_\sim$. Moreover, $(M_\sim,p')$ $< (M,p)$ for all $p' \in Pred_\sim$ and definitions (M,p) other than $(M_\sim,p)$.

An *extended program with rule prioritization* (*EPP*) is naturally translated into a *PMP* by considering each rule as a module that imports the predicates appearing in its body and exports its head predicate. Thus, we have also defined the reliable semantics for EPPs. The *RS* for *EPPs* extends the *well-founded semantics* for normal programs [11] and *extended well-founded semantics* for extended programs [7].

5. Related Work

The *contradiction removal semantics* (*CRS*) for extended programs [8] avoids contradictions brought about by closed world assumptions. For example, the *CRS* of $P=\{\neg p \leftarrow \sim a.\quad p \leftarrow.\quad b \leftarrow.\}$ is $\{p,\ b\}$ which is non-contradictory. Yet, contradictions not based on closed world assumptions cannot be resolved. For example, nothing is concluded from $P' = \{\neg p \leftarrow.\quad p \leftarrow.\quad b \leftarrow.\}$ though *b* should be true. The same is true for the semantics in [3, 12]. However, the $RM_{P'} = \{\}$.

The *conservative reasoning* for extended programs, presented in [13], is as follows: if *r* is a rule and $Body_r$ is true then $Head_r$ is true iff for every rule *r'* s.t. $Head_{r'}$ $= \neg Head_r$, $Body_{r'}$ cannot be derived. For example, the conservative semantics of $P =$ $\{r_1\!: \neg a \leftarrow b.\quad r_2\!: a.\quad r_3\!:b.\}$ is $\{b\}$. In *RS*, r_3 is considered unreliable and $RM_{P'} = \{\}$.

Prioritization of rules is investigated in the various variations of *ordered logic* [4, 6]. Even though these semantics are defined for all ordered logic programs, negation by default is not supported. Moreover, the rule ordering in ordered logic represents *exceptions* and not *reliability*. For, example, the ordered logic semantics of $P = \{r_1\!:$ $\neg a \leftarrow b.\quad r_2\!: a.\quad r_3\!:b.\}$ with $r_3 < r_2 < r_1$ is $\{b\}$ where as $RM_{P'} = \{a,\ \sim\neg a\}$ since $r_3 < r_2$. When the prioritization of the rules is ignored, ordered logic and conservative reasoning [13] behave similarly. Prioritization of rules is also supported in [2]. However, there rules are considered to be clauses, i.e., there is no distinction between the head and the body of a rule. Thus, in [2], program $P' = \{p \leftarrow \sim p.\}$ is considered equivalent with $\{p.\}$ whereas the *WFM* of *P'* is $\{\}$. The semantics of the above program *P* according to [2] coincides with $RM_{P'}$.

In [10], local *DBs* are combined with a supervisory *DB* in a framework based on *annotated logic*. However, though the supervisory *DB* can access literals defined in the local *DBs*, local *DBs* can access only local information. The resolution of conflicts between the local *DBs* is the responsibility of the supervisory *DB*.

6. Conclusions

We have presented the reliable semantics (*RS*) of *prioritized modular logic programs* (*PMPs*). The purpose of the reliable semantics is to derive reliable information from contradictory *PMPs*. Every *PMP* has at least one *stable m-model*. The *reliable model* of a program P is the least (w.r.t. $\subseteq$) stable m-model of P and it represents the skeptical "meaning" of P. Maximal (w.r.t. $\leq_{\mathrm{sat}}$) stable m-models of P represent the credulous "meanings" of P.

REFERENCES

[1] A. Bossi, M. Gabbrielli, G. Levi, M.C. Meo, "Contributions to the Semantics of Open Logic Programs", Proc. of the Intern. Conference of Fifth Generation Computer Systems (FGCS'92), 1992, pp. 570-580.

[2] S. Brass, U.W. Lipeck, "Semantics of Inheritance in Logical Object Specifications", 2nd Intern. Conf. on Deductive and Object-Oriented Databases, 1991.

[3] P.M. Dung, "An Argumentation Semantics for Logic Programs with Explicit Negation", 10th Intern. Conf. on Logic programming (ICLP'93), 1993, pp.616-630.

[4] D. Gabbay, E. Laenens, D. Vermeir, "Credulous vs. Sceptical Semantics for Ordered Logic Programs", Proc. of the 2nd International Conference on Knowledge Representation and Reasoning (KR'91), 1991, pp. 208-217.

[5] M. Gelfond, V. Lifschitz, "The Stable Model Semantics for Logic Programming", Proc. of the 5th Symposium on Logic Programming, 1988.

[6] E. Laenens, D. Vermeir, "Assumption-free Semantics for Ordered Logic Programs: On the Relationship Between Well-founded and Stable Partial Models", Journal of Logic and Computation, 2(2), 1992, pp. 133-172.

[7] L.M. Pereira, J.J. Alferes, "Well-founded Semantics for Logic Programs with Explicit Negation", 10th European Conf. on Artificial Intelligence, 1992, pp. 92-96.

[8] L.M. Pereira, J.N. Aparicio, J.J. Alferes, "Nonmonotonic Reasoning with Logic Programming", Journal of Logic Programming, 17, 1993, pp. 227-263.

[9] T. Przymusinski, "Well-Founded Semantics Coincides with the Three-Valued Stable Semantics", Fundamenta Informaticae XIII, 1990, pp. 445-463.

[10] V.S. Subrahmanian, "Amalgamating Knowledge Bases", to be published in ACM Transactions on Database Systems (TODS), 1994.

[11] A. van Gelder, K.A. Ross, J.S. Schlipf, "The Well-Founded Semantics for General Logic Programs", Journal of the ACM, 38(3), July 1991, pp.620-650.

[12] C. Witteveen, "Expanding Logic Programs", D. Pearce and G. Wagner (eds). Logics in AI, LNCS 633, 1992, pp. 372-390.

[13] G. Wagner, "Reasoning with Inconsistency in Extended Deductive Databases", Proc. of the 2nd Intern. Workshop on Logic Programming and Non-monotonic Reasoning, 1993, pp.300-315.

LaTeR: a general purpose manager of temporal information*

V. Brusoni[1], L. Console[1], B. Pernici[2], P. Terenziani[1]

[1] Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
[2] Dipartimento di Elettronica e Informazione. Politecnico di Milano,
Piazza Leonardo da Vinci 32, 20122 Milano, Italy

Abstract. LaTeR is a manager of temporal information which can be used in a loosely-coupled way by different types of applications. LaTeR is based on a layered architecture: at the physical layer heterogeneous (qualitative and quantitative) temporal information is amalgamated using a constraint framework and temporal reasoning is performed; the interface layer provides a high-level language for manipulating and querying temporal information. The expressive power of the language has been defined in such a way that complete temporal reasoning is tractable. Particular attention has been devoted to the design of the query language and to efficient query processing. Since in LaTeR temporal reasoning is performed during data manipulation (insertion and/or deletion of information), the complexity of answering a query depends only on the length of the query and not on the dimension of the knowledge base.

1 Introduction

A lot of attention has been paid in the A.I. community to the problem of dealing with time [2, 16]. In particular, most artificial intelligence approaches focus on reasoning issues [1, 10, 14, 15]. On the other hand, the problems of (i) designing a high level language for manipulating and querying temporal knowledge bases and (ii) answering queries efficiently have been often neglected. Some of these problems have been faced in the database community [9, 11, 12] where, however, reasoning and complexity issues received a limited attention.

The aim of this paper is to reconcile these two complementary tendencies in a general-purpose manager of temporal information: LaTeR (LAyered TEmporal Reasoner). LaTeR is conceived as a temporal knowledge server that can be used in a loosely-coupled way by different types of applications (e.g., artificial intelligence applications or temporal databases). In order to achieve such a goal, LaTeR is designed as a two-layer architecture: at the *physical layer* heterogeneous temporal information (possibly imprecise qualitative and quantitative information) is represented in a uniform way using a constraint framework; the

* This work was partially supported by CNR, project "Ambienti e strumenti per la gestione di informazioni temporali".

interface layer provides a high level language for manipulating and querying a temporal knowledge base. We believe that it is important that heterogeneous temporal information is reprented in a uniform way in the system (unlike, e.g., [7]). The uniform representation is in fact the basis of the reasoning process. As regards temporal reasoning, we claim that a knowledge server must operate in a predictable and efficient way. There are many aspects that contribute to a predictable behavior. First of all, correct and complete temporal reasoning must be performed. In case of incomplete reasoning (as, e.g., in TMM [5]), the answers provided by the system are, in general, unpredictably incomplete and users cannot safely rely on them (notice that in most cases users cannot have expectations on the result of combining a set of constraints). As a second aspect, complete reasoning must be tractable. This raises a trade off with respect to the expressive power of the representation language. It is well known that, in general, complete temporal reasoning is not tractable (e.g., complete reasoning in Allen's interval algebra [1] is intractable [15]) and thus limitations to the expressive power have to be introduced in order to achieve tractability. As a further aspect the manager must provide a powerful and clear language for manipulating and querying a knowledge base. Finally, query answering must be performed very efficiently (this is critical, e.g., in temporal databases).

LaTeR has been designed taking into account the goals discussed above. We analysed how heterogeneous temporal information can be represented in a uniform way starting from the notion of *distance* between time points. Such a notion can be mapped onto well known constraint satisfaction frameworks. We analysed precisely which types of information can be dealt with by a complete reasoner in polynomial time (section 2) and we designed a high-level language providing an interface to the internal constraint-based representation. Section 3 briefly outlines the primitives for manipulating and querying a temporal knowledge base. Special attention has been paid to the query language which provides modal operators and facilities for retrieving information from a knowledge base. We analysed how various types of queries can be answered efficiently. We showed that in our language if constraint propagation is performed when a knowledge base is built or updated, the complexity of query answering depends only on the dimension of the query and not on the dimension and content of the knowledge base (section 4).

2 Integrating Heterogeneous Temporal Information

Different assumptions can be made on the nature of time. As in many AI approaches, we assume time to be linear, totally ordered, metric and dense (i.e., time is isomorphic to real numbers). Time points are the basic temporal entities; an interval I is defined as a convex set of time points with a starting and an ending point, denoted respectively as $start(I)$ and $end(I)$ (with $start(I) < end(I)$).

Temporal information may be qualitative, quantitative or mixed qualitative-quantitative. Qualitative information concerns the relative position of temporal entities, as in (Ex.1) "Mary arrived before John started to work". Quantitative information either locates punctual and/or durative situations, as in (Ex.2) "John worked since 8^{00} until 12^{30}" or states the duration of durative situations, as in (Ex.3) "John worked for two hours". Mixed information provides delays, as in (Ex.4) "Mary arrived 30 minutes before John started to work". All these types of information may be specified with different degrees of precision: temporal information may be given precisely (as in the examples above), imprecisely, as in (Ex.5) "John started to work between 8^{00} and 8^{30}" or even in a disjunctive way, as in (Ex.6) "John worked either from 8^{00} until 12^{00} or from 8^{15}-8^{20} until 12^{15}-12^{20}". In order to provide a uniform primitive for amalgamating these different types of temporal information we introduce the basic notion of **distance between time points**:

Definition 1. Given two time points P_1 and P_2, distance(P_1,P_2,$[a, b]$) is true iff the distance between P_1 and P_2 is between a and b, where $a, b \in \Re$ and $a \leq b$.[3]

In the following we sketch how different types of temporal information can be mapped into formulae involving conjunctions and/or disjunctions of distances.

Any piece of precise or imprecise **quantitative information** can be mapped into a conjunction of distances. For instance, Ex.2 can be represented as :

$$\text{distance}(\mathit{start}(I_1),\ P_O,\ [8^{00}, 12^{30}]) \wedge \text{distance}(\mathit{end}(I_1),\ P_O,\ [12^{00}, 12^{30}])$$

(where I_1 is the time interval in which John worked). Here and in the following P_O represents the reference time for the temporal knowledge base, and corresponds to the 0^{00} of a given day. As an example of imprecise information, Ex.5 can be represented as:

$$\text{distance}(\text{start}(I_1),\ P_O,\ [8^{00}, 8^{30}])$$

On the other hand, disjunctive information also requires the introduction of disjunctions in the formulae. For instance, Ex.6 can be represented as follows:

$$(\text{distance}(\text{start}(I_1),\ P_O,\ [8^{00}, 8^{00}]) \wedge \text{distance}(\text{end}(I_1),\ P_O,\ [12^{00}, 12^{00}])) \vee$$
$$(\text{distance}(\text{start}(I_1),\ P_O,\ [8^{15}, 8^{20}]) \wedge \text{distance}(\text{end}(I_1),\ P_O,\ [12^{15}, 12^{20}]))$$

As regards **qualitative information** different algebrae of temporal relations have been proposed for dealing with time intervals [1], time points [15] or both of them [14]. Each algebra is based on a set of primitive relations. These relations can be easily mapped into conjunctions of distances (see, e.g., Table 5 as regards the interval algebra). In case knowledge about the relation between two entities is imprecise, ambiguous relations can be specified as disjunctions of basic relations and thus they can be mapped onto disjunctions of conjunctions of distances. In particular, three different cases can be distinguished [16]:

(1) Continuous pointisable relations are those relations between intervals or points and intervals which can be expressed as conjunctions of (possibly ambiguous) relations between their starting and ending points, without using inequality. These relations can be represented as conjunctions of distances, each one representing a (possibly ambiguous) relation between a pair of points. For instance, I_1 (STARTS $\vee$ DURING $\vee$ FINISHES $\vee$ EQUAL) I_2 can be represented as

$$\text{distance}(\text{start}(I_2),\ \text{start}(I_1),\ [0, \infty))) \wedge \text{distance}(\text{end}(I_2),\ \text{end}(I_1),\ [0, \infty)))$$

[3] The range $[a, b]$ may also be open at one or both of its extremes.

(2) Pointsable relations are those relations between intervals or points and intervals which can be expressed as conjunctions of (possibly ambiguous) relations between their starting and ending points, using also inequality. In such a case, a disjunction of distances (concerning the same pair of time points) must be introduced in order to represent each inequality.

(3) Non-Pointsable relations are those relations between intervals or points and intervals which cannot be expressed as a conjunction of (possibly ambiguous) relations between their starting and ending points. In such cases, disjunctions of conjunctions of differences (involving different pairs of time points) must be introduced. For instance, I_1 (BEFORE $\vee$ AFTER) I_2 can be represented as distance(start(I_2), end(I_1), $(0, -\infty)$)) $\vee$ distance(end(I_2), start(I_1), $(0, -\infty)$))

Mixed Qualitative and Quantitative information (i.e., delays between punctual situations) can be easily represented in terms of distances.

The notion of distance we use is isomorphic to the notion of difference between real numbers. If we associate two variables X_1 and X_2 with the time points P_1 and P_2, the primitive "distance(P_1,P_2,$[a,b]$)" can be represented as the *bound on difference* constraint "$a \leq X_2 - X_1 \leq b$". Thus temporal reasoning can be reduced to constraint propagation on bounds on differences [4, 6]. This provides a basis for analysing the complexity of temporal reasoning and the trade-off between expressive power and computational complexity.

Different classes of constraint satisfaction frameworks have been defined considering different ways of combining bounds on differences [6]. In STP *(Simple Temporal Problems)* only conjunctions of bounds on differences are allowed. In TCSP *(Temporal Constraints Satisfaction Problems)* also disjunctions of bounds on differences on the same pair of variables are allowed, i.e., we have conjunctions of disjunctions of bounds on differences and in which each conjunct involves only one pair of variables. In *Multiple* STP any form of disjunction (and conjunction) of bounds on differences is allowed. If such constraints are put into disjunctive normal form, then each disjunct can be regarded as an STP framework.

As regards the computational complexity, we have that in the case of STP, complete constraint propagation is performed in $O(n^3)$ steps, where n is the number of variables [4, 6]. In the other two cases complete constraint propagation is intractable (however, in the case of TCSP, specialized algorithms have been proposed, see e.g., [6]). Table 1 classifies various types of temporal information according to the constraint frameworks that can deal with them.

3 Architecture of LaTeR

In order to achieve the goals pointed out in Section 2, LaTeR has been designed as a layered architecture. At the physical layer heterogeneous temporal information is represented in a uniform way starting from the notion of distance; the layer is then implemented using STP. The interface layer provides a high level language for manipulating and querying a temporal knowledge base.

		STP	TCSP	Multiple STP
Location (point)	exact	X	X	X
	imprecise	X	X	X
	disjunctive		X	X
Location (interval)	exact	X	X	X
	imprecise	X	X	X
	disjunctive			X
Duration (interval)	exact	X	X	X
	imprecise	X	X	X
	disjunctive		X	X
Qualitative relations	Continuous	X	X	X
	Pointsable		X	X
	All			X
Delay (points)	exact	X	X	X
	imprecise	X	X	X
	disjunctive		X	X

Table 1. Expressive power of constraint frameworks. The labels "Continuous", "Pointsable" and "All" indicate Continuous Pointsable relations, Pointsable relations and the set of all the qualitative relations respectively.

3.1 Manipulation language

Temporal entities are entered in LATER via a declaration, in which the user specifies whether they are punctual or durative. The user can refer to the starting and ending point of a durative situation I as $start(I)$ and $end(I)$ respectively.
Quantitative relations. The constructs for locating points and intervals and their equivalent representation in terms of distances are shown in Tables 2 and 3 (in the tables P_O is the reference time point and a and b represent dates in the form *month/day/year hour:minute*). LATER also provides facilities for dealing with different levels of time granularity and calendar dates (however, these are not described in the paper for the sake of brevity). For instance,

I ATLEAST SINCE 1/1/1994 12:30 UNTIL 1/2/1994 12:30

states that the interval I lasts at least from 1/1/1994 12^{30} to 1/2/1994 12^{30}.

The construct LASTING (Table 4) allows one to specify the duration of a time interval (these will be expressed in the form *days:hours:minutes*). For instance,

I LASTING 1:23:59

states that the interval I lasts exactly 1 day, 23 hours and 59 minutes.
Qualitative relations. Since we deal with time points and time intervals, the interface language provides the following set of relations:

- 3 basic relations of the point algebra (i.e. $<,=,>$), plus the ambiguous relations ($\leq$, $\geq$) [15];
- 13 basic relations of the interval algebra [1], see Table 5;
- 10 basic relations of the point/interval algebra [14].

Notice that the possibility of specifying relations between the starting/ending points of intervals (accessed using the the *start* and *end* constructs) allows us to specify any continuous pointisable relation between intervals and between points and intervals. Moreover, the system provides the NON-STRICT construct for some basic relations of the interval algebra and point/interval algebra, to deal with some of the most commonly used ambiguous relations; for instance, the relation I_1 NON-STRICT DURING I_2 denotes non-strict inclusion, i.e., the ambiguous relation I_1 (STARTS $\vee$ DURING $\vee$ FINISHES $\vee$ EQUAL) I_2.

Mixed Relations. Delays between time points can be specified by associating metric information with the constructs BEFORE and AFTER, as in:

P1 ATMOST 0:1:0 BEFORE P2

which states that the point P1 happens at most 1 hour before P2. The translation in terms of distances is straightforward.

Interface language	**Mapping to distances**
P AT a	distance(P, P_o, [a,a])
P BETWEEN a AND b	distance(P, P_o, [a,b])
P AFTER a	distance(P, P_o, (a,+∞))
P BEFORE b	distance(P, P_o, (−∞,b))

Table 2. Location of a time point P.

Interface language	**Mapping to distances**
I SINCE a UNTIL b	distance(*start*(I), P_o, [a,a]) $\wedge$ distance(*end*(I), P_o, [b,b])
I SINCE a-b UNTIL c-d	distance(*start*(I), P_o, [a,b]) $\wedge$ distance(*end*(I), P_o, [c,d])
I ATMOST SINCE a UNTIL b	distance(*start*(I), P_o, [a,b]) $\wedge$ distance(*end*(I), P_o, [a,b])
I ATLEAST SINCE a UNTIL b	distance(*start*(I), P_o, (-∞,a]) $\wedge$ distance(*end*(I), P_o, [b,+∞))

Table 3. Location of a time interval I.

Interface language	**Mapping to distances**
I LASTING d	distance(*end*(I), *start*(I), [d,d])
I LASTING ATLEAST d	distance(*end*(I), *start*(I), [d,+∞))
I LASTING ATMOST d	distance(*end*(I), *start*(I), (0,d])

Table 4. Duration of an interval I.

Interface language	Mapping to distances
I_1 BEFORE I_2	distance($start(I_2)$,$end(I_1)$,(0,+∞])
I_1 MEETS I_2	distance($start(I_2)$,$end(I_1)$,[0,0])
I_1 OVERLAPS I_2	distance($start(I_2)$,$start(I_1)$,(0,+∞]) ∧ distance($end(I_1)$,$start(I_2)$,(0,+∞]) ∧ distance($end(I_2)$,$end(I_1)$,(0,+∞])
I_1 DURING I_2	distance($start(I_1)$,$start(I_2)$,(0,+∞]) ∧ distance($end(I_2)$,$end(I_1)$,(0,+∞])
I_1 FINISHES I_2	distance($start(I_1)$,$start(I_2)$,(0,+∞]) ∧ distance($end(I_2)$,$end(I_1)$,[0,0])
I_1 EQUAL I_2	distance($start(I_1)$,$start(I_2)$,[0,0]) ∧ distance($end(I_2)$,$end(I_1)$,[0,0])

Table 5. Basic relations in Allen's Interval Algebra. For the sake of brevity, "inverse" relations have been omitted

3.2 Query language

Temporal queries can be expressed using the same operators used for data manipulation, plus two modal operators. In fact, since temporal information may be imprecise (e.g., P BETWEEN a AND b), it is necessary to distinguish if some conclusion must necessarily hold, or if it is only consistent with the temporal database (see also the discussion in [8] and [13]). For instance, let us suppose that the database contains only the following assertion:

P BETWEEN 12:00 AND 13:00

Consider now the following queries:

does P happen at 12^{30} *?*
does P happen between 11^{00} *and* 14^{00} *?*
does P happen at 14^{00} *?*

Given the first query, it is possible (consistent with the temporal database) that P happens at 12^{30}, but this is not certain. Given the second one, it is certain that P happens between 11^{00} and 14^{00}, since it is constrained by the database to happen between 12^{00} and 13^{00}. The query language provides two modal operators to distinguish between two types of queries:

- queries asking whether something is necessarily true, for instance:

  MUST P AT 12:30 ⇒ Answer: NO
  MUST P BETWEEN 11:00 AND 14:00 ⇒ Answer: YES
  MUST P AT 14:00 ⇒ Answer: NO

- queries asking whether something is consistent with the database, for instance:

  MAY P AT 12:30 ⇒ Answer: YES
  MAY P BETWEEN 11:00 AND 14:00 ⇒ Answer: YES
  MAY P AT 14:00 ⇒ Answer: NO

The modal operators can be applied to any conjunction of sentences of LATER's manipulation language to form queries. Moreover, it is possible to place variables in the queries to retrieve pieces of temporal information from the database; for instance, if the database contains only the assertion

I SINCE 12:00 UNTIL 13:00

it is possible to ask the following queries:

MAY I LASTING X? ⇒ Answer: X? = 0:1:0

MUST (start(I) At X? AND end(I) At Y?) ⇒ Answer: X?=12:00, Y?=13:00

4 Temporal Reasoning in LATER

Temporal information is introduced in LATER using the high-level manipulation language presented in section 3.1. Each assertion is translated in constant time into the corresponding conjunction of distances and then into a set of bounds on differences on which constraint propagation is performed. The fact that we limit to conjunctions of differences guarantees that we can use an STP framework.

The result of constraint propagation in STP is the set of *maximal admissibility ranges* for the differences between each pair of variables. This is defined as follows: the *maximal admissibility range* for the difference $X-Y$ is the maximal interval $[a,b]$ which includes all and only the values v for $X-Y$ satisfying all the constraints (i.e., such that for each $v \in [a,b]$ the original set of constraints with the assignment $X-Y=v$ is consistent). Notice that if one is interested in the maximal admissibility range for a variable X, it is sufficient to add a variable X_0 representing a reference value for the whole knowledge base and then consider the maximal admissibility range for the difference $X-X_0$. Thus the set of maximal admissibility ranges for all pairs of variables characterize in a compact way all the solutions to a constraint satisfaction problem. A set of bounds on differences is inconsistent if and only if there is a pair of variables X and Y such that the maximal admissibility range for the difference $X-Y$ is $[a,b]$ with $b<a$.

In LATER we chose to perform constraint propagation when a knowledge base is manipulated. Maintaining the propagated knowledge base (i.e., the set of maximal admissibility ranges) has important advantages as regards the complexity of answering queries (at least in those cases where query processing is more frequent than the updating of the knowledge base). In fact, in such a case the complexity of query answering depends only on the length of the query and not on the dimension and content of the knowledge base (see [3]).

Given a query $Q \equiv OP\ EXPR$ (where OP is one of the $MUST$ or MAY operator and $EXPR$ is a conjunction of LATER assertions – see section 3.2), LATER operates as follows: (1) $EXPR$ is translated into the corresponding set S of bounds of differences; (2) the query $OP\ S$ is processed at the physical level. Two cases have to be distinguished: (1) S is a singleton, (2) S is a set of bounds on differences.

In case S is a singleton the answer can be provided in constant time with a simple lookup into the propagated knowledge base. More specifically, let us suppose that $[a, b]$ is the maximal admissibility range for the difference $X - Y$ and that $S \equiv c \leq X - Y \leq d$:

- $MUST(c \leq X - Y \leq d) \Leftrightarrow [c, d] \supseteq [a, b]$
  Intuitively, since the maximal admissibility range $[a, b]$ includes *all* the values for $X - Y$ satisfying the constraints in the knowledge base, then a-fortiori any interval $[c, d]$ such that $[c, d] \supseteq [a, b]$ includes all the values for $X - Y$ satisfying all the constraints in the knowledge base.
- $MAY(c \leq X - Y \leq d) \Leftrightarrow [c, d] \cap [a, b] \neq \emptyset$
  Intuitively, since the maximal admissibility range $[a, b]$ includes *only* values for $X - Y$ satisfying the constraints in the knowledge base, then any interval $[c, d]$ that intersects $[a, b]$ contains at least one value for $X - Y$ satisfying all the constraints in the knowledge base.

In case S is a set $C_1, \ldots, C_n$, the complexity of answering queries is different in case of $MUST$ and MAY queries. In particular:

- In case of $MUST$ queries, the $MUST$ operator can be distributed, i.e.:

$$MUST(\{C_1, \ldots, C_n\}) \Leftrightarrow MUST(C_1) \wedge \ldots \wedge MUST(C_n)$$

  Thus the complexity of answering the query is $O(n)$.
- MAY is a modal possibility operator and does not distribute over conjunctions and thus the constraints in S cannot be checked independently of each other. Indeed, a query MAY S corresponds to checking that S is consistent with the knowledge base. This can be done only by adding the constraints in S to the knowledge base and performing constraint propagation. However we proved that propagation to the whole knowledge base is not needed and it is sufficient to consider the variables occurring in S, that is (see [3]):

  Theorem. *Given a knowledge base K.B., a query MAY S and two variables X and Y is S, the maximal admissibility range for $X - Y$ obtained with global propagation of the constraints in S to all the variables in K.B. is the same as the the maximal admissibility range for $X - Y$ obtained with local propagation of the constraints in S to the variables in S.*

  This means that we can answer a query $MAY(\{C_1, \ldots, C_n\})$ with local propagation to the variables in $C_1, \ldots, C_n$ and thus the complexity is $O(n^3)$, regardless of the dimension and content of the knowledge base.

The theorem above guarantees that also queries with variables can be answered with local propagation and thus independently of the dimension of the knowledge base.

5 Discussion

In the paper we presented LATER, a general-purpose and modular manager of heterogeneous temporal information. LATER compares favourably to other approaches. First of all, in LATER heterogeneous information is represented in a uniform way (unlike, e.g., [7]) and complete temporal temporal reasoning is performed (unlike, e.g., [5] which has basically the same expressive power). These two features have been taken into account in Meiri's approach [10], who, however, did not analyse query answering issues. The problem of query processing has been tackled by Van Beek [13], who introduced the same modal operators used in LATER. However, in Van Beek's approach, given any query, constraint propagation is performed on the whole knowledge base and thus the complexity is cubic in the dimension of the knowledge base (Van Beek allows also the use of disjunction in the queries but this makes query answering intractable).

A prototype of LATER has been implemented in Prolog on Macintosh.

References

1. J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.
2. J. Allen. Time and time again: the many ways to represent time. *Int. J. Intelligent Systems*, 6(4):341–355, 1991.
3. V. Brusoni, L. Console, B. Pernici, and P. Terenziani. On the computational complexity of query answering in bounds on differences constraints. Technical report, Dip. di Informatica, Università di Torino, Torino, 1993.
4. E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.
5. T. Dean and D. McDermott. Temporal data base management. *Artificial Intelligence*, 32:1–56, 1987.
6. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
7. H. Kautz and P. Ladkin. Integrating metric and qualitative temporal reasoning. In *Proc. AAAI 91*, pages 241–246, 1991.
8. R. Maiocchi, B. Pernici, and F. Barbic. Automatic deduction of temporal information. *ACM Trans. on Database Systems*, 17:647–688, 1992.
9. L. McKenzie and R. Snodgrass. Evaluation of relational algebras incorporating the time dimension in databases. *ACM Computing Surveys*, 23(4):501–543, 1991.
10. I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In *Proc. AAAI 91*, pages 260–267, 1991.
11. R. Snodgrass, editor. *Proc. of the Int. Work. on an infrastructure for Temporal Databases.* 1993.
12. A. Tansell, R. Snodgrass, J. Clifford, S. Gadia, and A. Segev. *Temporal Databases: Theory, design and implementation.* Benjamin Cummings, 1993.
13. P. VanBeek. Temporal query processing with indefinite information. *Artificial Intelligence in Medicine*, 3:325–339, 1991.
14. M. Vilain. A system for reasoning about time. In *Proc. AAAI 82*, pages 197–201, 1982.
15. M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. AAAI 86*, pages 377–382, 1986.
16. M. Vilain, H. Kautz, and P. VanBeek. Constraint propagation algorithms for temporal reasoning: a revised report. In D.S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about physical systems*, pages 373–381. Morgan Kaufmann, 1989.

Understanding a story with causal relationships

Honghua Gan*

Department of Computer Science, University of Exeter, Exeter EX4 4PT, England,
hga@uk.ac.exeter.dcs

Abstract. In our previous work (e.g., [4], [5], [6], [7]), we have formalized the story understanding process based on scripts and plans with stepwise default theories. While those theories offer final results for understanding a specific story, they do not provide the history of changes of partial states of any objects the story may concern. Moreover, the causal models for missing events are incomplete in script-based understanding, and even not involved in plan-based understanding. As the result, the understanding process lacks the causal foundation. In this paper, we propose a default rule representation of causal relationships. In common sense situations, we give an event-based analysis for this general representation to fix its structure. A complete causal model for a story, i.e., a default causal chain, is developed for understanding the story. Stepwise default theories and frame-based systems are described. The latter provides the history of partial state changes of agents and objects in the story by generating an understanding chain.

1 Introduction

According to [16], people understand a story through reminding similar experiences in the past. A typical form of such experiences is a script. In [15], script-based understanding and plan-based understanding have been discussed, which are further formalized with default theories in [4], [5], [6] and [7]. While those theories offer final understandings of a story, they do not provide the whole history of changes of partial states of agents and objects in the story. Moreover, the causal model (as analyzed in conceptual dependency primitives) are incomplete in script-based understanding, and even not involved in plan-based understanding, so that the understanding process lacks an appropriate causal foundation.

In the literature, a number of logical formalizations have been proposed to represent causal knowledge like "X causes Y" (e.g., [10], [19], [1], [20]). In AI research, some practical causal reasoning models have also been created (e.g., [11], [13], [14], [2], [9]). As far as nonmonotonicity is concerned, most of those formalizations and causal models fail to capture the nature of causal relationships ([7]). Even Shoham's causal theory [17], which is developed from his nonmonotonic

* The author is supported by Sino-British Friendship Scholarship Scheme (SBFSS) and Natural Science Foundation of China (NSFC).

framework, is not an appropriate representation for causal relationships because of odd interpretations for modal operators ([3]).

We propose a default rule representation for causal relationships ([7]). A typical statement that X causes Y can be simply described as $X : A/Y$, where X is the cause, Y the effect and A the causal fields which together with X make the occurrence of Y. This representation employs a number of advantages over other existing theories and meets characteristics of causal relationships somehow: (1) It includes other causal factors A besides X and Y, and A as justifications indeed is less important than X for the occurrence of Y. It reflects a normal way to jump from X to Y when A is not violated. Reasoning about it is nonmonotonic because new information if violates A would block the default rule and hence Y would no longer be inferred. It is context-sensitive because A describes different contexts. (2) It describes propositions in their plain forms based on facts happening in the real world. A simple and clear interpretation of causal relationships is provided, that is, if X occurs and A is normal then Y is expected to occur. The oddness of modal operators of causal epistemic semantics (as in [1] or [17]) is avoided. (3) It satisfies antireflexivity, antisymmetry, and non-transitivity in general; it is not a simple logical implication. It may be able to express other kinds of causal knowledge[2]. Its reasoning mechanism is basically for causal prediction.

2 Event-based analysis for causal relationships

In common sense situations where a specific story happens, we have to further fix the structure of causal relations. That is, constructions of X, A, and Y in $X : A/Y$ should be analyzed with respect to the specific structure in a story. We suppose that facts happening in the real world are isolated events. Causal relationships are at the deep level to fill in gaps between those fact level isolated events. After understanding a story, we can have a result where the story becomes a causally well-connected sequence of events.

2.1 Event structure

We suppose there are a number of objects and agents in a common sense situation and agents can produce actions. An object (or agent) has many attributes, which in turn identify what the object means. An event explicitly describes in a natural language sentence a series of partial state changes of concerned agents and objects when an action occurs. A partial state is a particular existence with appropriate attribute and value where an agent or an object holds on. In structure, an event consists of some explicit pre-state, action and explicit post-state, when the action occurs, changes happen from explicit pre-state to explicit post-state, which can be described by a logical implication in predicate calculus. For example, the following event "John was hungry, he entered the restaurant" can be represented as

[2] For example, a causal prevention "X prevents Y" could be represented as $X : A/\neg Y$ where X is the cause, $\neg Y$ the effect and A the causal fields.

$$S_{John}(John, \langle(hungry, yes)\rangle) \wedge Action(John, *enter, Restaurant) \rightarrow$$
$$S'_{John}(John, \langle(where, inside)\rangle).$$

2.2 Causal relation within events

Because of the structure of events, the causal relation within events must be considered with pre-state, action and post-state as well although here pre-state and post-state should include both explicit and implicit partial states since a causal relation is seen as the deep level knowledge. There still exist many possible ways to map these partial states and action into a default rule ([7]). We here propose a mapping such that one particular partial state in pre-state is the cause (prerequisite), action and other partial states in pre-state are the causal fields (justifications) and post-state is the effect (consequent)[3].

According to our mapping, the causal relationship underlying the above event should be "John being hungry causes him to enter the restaurant under the condition that the restaurant is open for business, John is outside the restaurant, he has some money, etc.".

$$\begin{gathered} Ps_{John}(John, \langle(hungry, yes)\rangle) : \\ Ps_{John}(John, \langle(money, M), (where, outside)\rangle) \wedge \\ Ps_{Restaurant}(Restaurant, \langle(business, open), (customers, N)\rangle) \wedge \\ Action(John, *enter, Restaurant)/ \\ Ps_{John}(John, \langle(hungry, yes), (where, inside), (money, M)\rangle) \wedge \\ Ps_{Restaurant}(Restaurant, \langle(business, open), (customers, N+1)\rangle). \end{gathered}$$

Clearly, its interpretation shares our common sense intuition. The general representation should be

$$EPS : (Pre.state - EPS) \wedge Action/Post.state$$

where EPS refers to a particular explicit partial state, $Pre.state - EPS$ is the set of the pre-state of the event excluding this particular explicit partial state.

2.3 Causal relation between events

Causation between events is basically more general than causation within events. Consider the two events "John sat down (e_1), the waiter came over with the menu (e_2)." Note that e_1 occurs before e_2, so that the causation actually exists from the post-state of e_1 to e_2. That is, John being sitting there causes the waiter to come over with the menu. The cause obviously belongs to the changed part from the pre-state to the post-state of e_1 after the first action occurs. Let the changed partial states be the set of $post.state_1 - pre.state_1$ and EPS_1 (i.e., John being sitting) $\supset post.state_1 - pre.state_1$. The default assumptions A need to be fixed

[3] The action may be seen as one of the cause (as one referee suggested). But here we do not think the action should be always explicitly described (it could be missing). It had better be only assumed by default.

further. In addition to the pre-state of e_2, A should also include the post-state of e_1 excluding EPS_1, because after the first action occurs, the post-state of the first event is also available as conditions for the occurrence of e_2. The default rule representation for causal relation between two events should be

$$EPS_1 : (Post.state_1 - EPS_1) \wedge Pre.state_2 \wedge Action_2 / Post.state_2,$$

which has the same form as the representation within events.

2.4 Default causal chain for the whole story

A story can be seen as a sequence of events. According to [15], understanding a story is not simply to understand isolated events separately. Instead, it is to make everything missing at the fact level to be figured out and make them causally well-connected. The deep level causal relation within each of them individually is not helpful for understanding the whole story. The causal relation between them may not be directly created because they are basically not causally involved (a number of events are missing between two isolated neighboring events).

Let two events e_1 and e_2 be neighboring events in the story. Some events are missing between them. According to the sequential order, these missing events can be explicitly described to form causal relations each other or with an event explicitly described at the fact level. For instance, there exists an event e_{1x} immediately before e_1 such that there is a causal relation between e_{1x} and e_1. Similarly, there exists an event e_{1y} immediately after e_1 such that there is a causal relation between e_1 and e_{1y}; and so on. These causal relations work together to make the isolated events in the story causally well-connected. This sequence of transitive causal relations[4] can be called a *default causal chain*, which is viewed as common sense knowledge at the deep level behind the whole story.

In general, let the motivation of the story happening in familiar situations indexing the default causal chain be MOT, which is also the cause of the first causal relation in the default causal chain. Transitive causal relations in the default causal chain, represented by default rules, are as follows:

$\delta_1 = MOT : (Pre.state_1 - MOT) \wedge Action_1 / Post.state_1,$
$\delta_2 = EPS_1 : (Post.state_1 - EPS_1) \wedge Pre.state_2 \wedge Action_2 / Post.state_2,$
$\ldots\ldots,$
$\delta_n = EPS_{n-1} : (Post.state_{n-1} - EPS_{n-1}) \wedge Pre.state_n \wedge Action_n / Post.state_n,$

where EPS_i is a particular partial state of $Post.state_i$ and one of the changed partial states from $Pre.state_i$ to $Post.state_i$. Because of event ordering in the chain, their causal applicable ordering may be: $\delta_1 < \delta_2 < \ldots < \delta_n$. Namely, δ_i would be used to fill in the potential gaps only after δ_{i-1} has already been used. The default causal chain of eating in the restaurant can be found in ([7]).

[4] Note that in general causal relations are non-transitive, but here there exist no conflicts between causal fields of these causal relations in a default causal chain.

3 Stepwise default theories for story understanding

Let the story consist of isolated events $e_1, e_2, \ldots, e_m$, Each of which is explicitly expressed as a fact,

$$e_1 = S_0^1 \wedge A_1 \rightarrow S_1^1,\ e_2 = S_0^2 \wedge A_2 \rightarrow S_1^2,\ \ldots,\ e_m = S_0^m \wedge A_m \rightarrow S_1^m,$$

where S_i^j are explicit states and A_i explicit actions. Let the default causal chain behind the story be $DCC = \{\delta_1 < \delta_2 < \ldots < \delta_n\}$. Stepwise default theories for causal prediction can be created

$$\Delta_1(DCC, F_1), \Delta_2(DCC, F_2), \ldots, \Delta_m(DCC, F_m),$$

such that

$$F_1 = \{e_1\}, F_2 = F_1 \cup \{e_2\}, \ldots, F_m = F_{m-1} \cup \{e_m\}.$$

Let $E_1, E_2, \ldots, E_m$ be extensions of the above stepwise default theories. As in script-based understanding ([4]), there are three cases about extensions:

1. Confirmation: $E_1 = E_2 = \ldots = E_m$. There is no conflict between partial states or actions explicitly described in the story and the causes or the causal fields of causal relations in the default causal chain, as in "John was hungry, he entered the restaurant. He asked for a dish. He left."
2. Distraction: $E_1 = E_2 = \ldots = E_i, E_i \neq E_{i+1}, \ldots, E_i \neq E_j, E_i = E_{j+1} = \ldots = E_m$. That is, when the isolated event e_{i+1} comes along, its explicit partial state or action violates the cause or the causal fields of a certain causal relation in the default causal chain. This causal relation is blocked and the story digresses at this point from the default causal chain. But a later isolated event e_{j+1} reconfirms a causal relation in the default causal chain and the story comes back to the chain, as in "John was hungry, he entered the restaurant. He met Bill. They talked for a while. He asked for a dish. He left."
3. Interference: $E_1 = E_2 = \ldots = E_i, E_i \neq E_{i+1}, E_i \neq E_{i+2}, \ldots, E_i \neq E_m$. Again, when the isolated event e_{i+1} comes along, its explicit partial state or action violates the cause or the causal fields of a certain causal relation in the default causal chain. This causal relation is blocked, the story has gone away from the default causal chain at this point and will never come back to the chain, as in "John was hungry, he entered the restaurant. He met Bill. They went home together."

Stepwise default theories in general enjoy the simplicity of representation and concise semantic interpretation. But it loses inference structures from which the final result is derived. In other words, stepwise default theories only give results for causal prediction in understanding a story, and there is no intermediate results for causal explanation and causal diagnosis, both of which should be important tasks of story understanding. The contexts identified by causal prediction must be retained for causal explanation and causal diagnosis.

In addition, a rather technical problem in stepwise default theories concerns recomputation. Especially in the case of confirmation, all extensions are to be the same, but at each step all default rules are computed one by one. A more efficient representation is needed, where the computation process for intermediate results is not repeated, and more importantly these intermediate results can be retained with causal prediction. A frame-based system, as developed below, can meet this challenge.

4 Frame-based systems for story understanding

Traditionally, frame-based systems can deal with understanding some static information such as various references to the same object, static facts acquired from the story and default properties ([5]). Objects obviously can be represented as frames with slots and values stored by properties. The whole default causal chain can be linked to the concerned object frame by *dcc* links, e.g., the default causal chain for eating in the restaurant will be associated to the object frame *RESTAURANT*. Such objects frame will be called *host frames*.

Causal relations in the default causal chain will also be formed as frames. These frames will be connected by *dcc* links, arranged by the applicable ordering between these causal relations. The translation from the sequence of default rules to the sequence of the *dcc* frames is straightforward. First of all, the host frame has some slots to store the motivation MOT and partial states in $Pre.state_1 - MOT$. A *dcc* frame df_i has slots to store $Action_i$ and partial states in EPS_i, $Post.state_i$ and $Pre.state_{i+1}$. Partial states in $Pre.state_i$ have been already stored in the previous *dcc* frame df_{i-1} or the host frame (accordingly the host frame can be written as df_0). Not all partial states in EPS_i and $Post.state_i$ need to be explicitly stored in df_i because a lot of partial states remain unchanged after applying a causal relationship and can be inherited from the previous *dcc* frames along the *dcc* links. Accordingly, partial states involved in the causal relation δ_i should be divided into three parts:

1. The unchanged partial states from $Pre.state_i$ to $Post.state_i$ after $Action_i$ occurs (IS_i). Such partial states are not to be explicitly expressed in df_i, because they can be obtained by inheritance along the chain.
2. The changed partial states from $Pre.state_i$ to $Post.state_i$ after $Action_i$ occurs (CS_i). Of course these partial states should be explicitly stored in df_i. EPS_i, a particular partial state of this kind, will be stored in the slot *cause*.
3. The new partial states $Pre.state_{i+1}$ which are the context of the next causal relation (NS_i). These partial states should also be stored in df_i explicitly (because of well-connectedness).

Such a frame-based system includes two types of frames: object frames and *dcc* frames. When isolated events in the story are received stepwise, relevant object frames need to be instantiated and an understanding chain will be produced by causal prediction. This understanding chain for the story provides the history of changes of all partial states and records all missing events through causal connections. Causal explanation and causal diagnosis can be done on those. Furthermore, information can be shared between *dcc* frames and relevant object frames. Both dynamic and static dimensions of understanding will be achieved.

Causal prediction in the frame-based system depends on whether a newly received event of the story matches a *dcc* frame in the default causal chain. Intuitively, an event e_i of the story matches a *dcc* frame df_j in the default causal chain if and only if (1) $S_0^i = EPS_{j-1}$, or there is no conflict between S_0^i and $(IS_{j-1} \wedge CS_{j-1} \wedge NS_{j-1})$[5]; (2) $S_1^i \supset CS_j$; and (3) there is no conflict between A_i and $Action_j$.

According to stepwise understanding, when e_i matches df_j (which is called a *matching point* and written as $MP(e_i)$), it actually confirms the implicit causal relations from $MP(e_{i-1})$ to $MP(e_i)$. Furthermore, if all events $e_1, e_2, \ldots, e_m$ in the current story match *dcc* frames respectively, then a partial sequence of *dcc* frames, e.g., $df_1, df_2, \ldots, df_k$ $(MP(e_m))$ in the default causal chain have been confirmed. Let the partial understanding chain at various stages be UC_i (say $UC_0 = df_0$), then its development process may be as follows:

$$UC_1 = UC_0 \leftarrow \ldots \leftarrow MP(e_1);\ UC_2 = UC_1 \leftarrow \ldots \leftarrow MP(e_2);\ \ldots;$$
$$UC_m = UC_{m-1} \leftarrow \ldots \leftarrow MP(e_m).$$

The other *dcc* frames $df_{k+1}, \ldots, df_n$ will be added to form the final understanding chain, i.e.,

$$UC = UC_m \leftarrow df_{k+1} \leftarrow \ldots \leftarrow df_n.$$

Here all *dcc* frames in the understanding chain are instantiated by the current story. This case corresponds to the case of confirmation as discussed in stepwise default theories for understanding. But here there is no repeat computation and the history of partial state changes are explicitly available for causal explanation and causal diagnosis.

On the other hand, If an event e_i does not match any frame in the default causal chain (corresponding to the case of distraction or interference as discussed in stepwise default theories for understanding), the current story will not follow the default causal chain from somewhere. That breaking point must be reported and retained in the understanding chain for causal diagnosis. Accordingly, an event e_i breaks the default causal chain at the frame df_j if and only if (1) df_j is not confirmed by the previous events; (2) e_i does not match any frame in the default causal chain; and (3) there is a pre-state conflict at df_{j-1}, or an action conflict at df_j, or a post-state conflict at df_{j-1}.

According to stepwise understanding, when e_i breaks the default causal chain at df_j (which is called a *breaking point* and written as $BP(e_i)$), it actually confirms the *dcc* frames from $MP(e_{i-1})$ to $BP(e_i)$. These frames should be added to the current partial understanding chain while the event e_i replaces df_j,

[5] If $j = 1$, then $EPS_0 = MOT$, IS_0 and CS_0 are undefined, but NS_0 can be defined as $Pre.state_1 - MOT$.

$$UC_i = UC_{i-1} \leftarrow \ldots df_{j-1} \leftarrow e_i.$$

In the case of distraction, when a later event e_r matches a *dcc* frame df_k ($k \geq j$) (which is called a recovering point), it actually confirms the *dcc* frames from $BP(e_i)$ (df_j) to df_k (maybe $j = k$). That is, the current partial understanding chain is

$$UC_r = UC_{r-1} \leftarrow df_j \leftarrow \ldots \leftarrow df_k.$$

Certainly, the events from e_i to e_r did not match any *dcc* frames in the default causal chain. This sequence of isolated events is called *an accident* to the default causal chain. The partial understanding chains at those stages may be

$$UC_{i+1} = UC_i \leftarrow e_{i+1}; \; UC_{i+2} = UC_{i+1} \leftarrow e_{i+2}; \ldots; \; UC_{r-1} = UC_{r-2} \leftarrow e_{r-1}.$$

Clearly, the accident is inserted between df_{j-1} and df_j in the partial understanding chain. There should exist another default causal chain behind this accident, and understanding this accident would produce an understanding chain for it, which can replace this accident here. Therefore the understanding chain for the current story is still causally well-connected although it may be multiple-level nested. Causal explanation and causal diagnosis can be carried out on it.

In the case of interference, no event after e_i in the current story will come back to rematch any frames in the default causal chain. The understanding chain will be formed as follows:

$$UC_{i+1} = UC_i \leftarrow e_{i+1}; \; UC_{i+2} = UC_{i+1} \leftarrow e_{i+2}; \ldots; \; UC_m = UC_{r-2} \leftarrow e_m.$$

This understanding chain is no longer causally well-connected from e_i. Implicit partial states and actions may not be completely described and causal explanation and causal diagnosis may not be done based on that.

Intermediate results in the understanding chain provide missing events and causal connections between these events through partial states and actions explicitly stored in the *dcc* frames of the understanding chain and implicitly inherited along the understanding chain. Inheritable partial states may be shared between relevant object frames and *dcc* frames. Inheritance through *isa* links along object frames can be formalized as follows

$$\begin{gathered}
hold(f_y, s_1, v_1) : isa(f_x, f_y) \wedge hold(f_x, s_1, v_1)/hold(f_x, s_1, v_1),\\
hold(f_y, s_2, v_2) : isa(f_x, f_y) \wedge hold(f_x, s_2, v_2)/hold(f_x, s_2, v_2),\\
\ldots,\\
hold(f_y, s_k, v_k) : isa(f_x, f_y) \wedge hold(f_x, s_k, v_k)/hold(f_x, s_k, v_k).
\end{gathered}$$

Here, f_x and f_y are object frames. The predicate *hold* indicates a partial state with the representation $Ps(o, \langle(at, v)\rangle)$ where o briefly corresponds to the frame, at reflects the slot, and v accounts for the value of the slot.

Similarly, inheritance via *dcc* links along the understanding chain can be formalized as follows:

$$hold(df_i, s_1, v_1) : dcc(df_{i+1}, df_i) \wedge hold(df_{i+1}, s_1, v_1)/hold(df_{i+1}, s_1, v_1),$$
$$hold(df_i, s_2, v_2) : dcc(df_{i+1}, df_i) \wedge hold(df_{i+1}, s_2, v_2)/hold(df_{i+1}, s_2, v_2),$$
$$\ldots,$$
$$hold(df_i, s_n, v_n) : dcc(df_{i+1}, df_i) \wedge hold(df_{i+1}, s_n, v_n)/hold(df_{i+1}, s_n, v_n).$$

Namely, if the precedent *dcc* frame df_i has stored some new partial states, those partial states would be unchangedly inherited by the *dcc* frame df_{i+1} that is linked to df_i by the *dcc* link, unless conflicts have been announced in the slot of changed partial states (CS).

5 Conclusions and improvements

Understanding with default causal chain provides a complete causal model at the deep level for a specific story. Both explicit and implicit partial states have been described in the model. Changes of any partial states are achieved by applying a causal relationship. The understanding process consists of three tasks: causal prediction, causal explanation and causal diagnosis. The idea is the intermediate results identified by causal prediction can be used for causal explanation and causal diagnosis. While stepwise default theories fail to provide such results, frame-based systems can do it by retaining them through inheritable understanding chain. Meanwhile, dynamic and static understanding are both captured in the frame-based system.

The concept of *default causal chain* proposed in the paper is similar to *script*. However, there are many significant differences, e.g., there are significant differences in originality and development, because scripts have been developed from past experiences in familiar situations ([8]), but a default causal chain is based on the causality although it is not clear whether and how the causality itself can be learned; there are significant differences in research methodology, because a script is emphasized as a default for understanding familiar situations while a default causal chain tries to start from causal relations, in which *default* is one of their properties; there are significant differences in structure, because conceptual dependency primitives do not support implicit information at the deep level while causal relationships represented both by default rules and frames, can not only fill in missing events and connections, but also provide implicit information for each events; there are significant differences in application, because script-based understanding deals with missing events and connections therefore basically a script is for dynamic dimensions of understanding while a default causal chain supports missing events, connections and implicit partial states and actions, and includes both dynamic and static understanding, because all changes of partial states of each object involved in the story will be described in the understanding process in addition to making missing events and connections explicit.

References

1. Arthur W. Burks: *Chance, Cause, Reason: An Inquiry into the Nature of Scientific Evidence*, The University of Chicago Press, 1977.

2. Johan de Kleer and John Seely Brown: *Theories of Causal Ordering,* Artificial Intelligence 24 (1986) 33-62.
3. Antony Galton: *A Critique of Yoav Shoham's Theory of Causal Reasoning,* Proceeding of IJCAI-91, 355-359.
4. Honghua Gan: *Formalizing Scripts with Default Theory,* Technical Report 248, University of Exeter, Department of Computer Science, 1992.
5. Honghua Gan: *Script and frame: mixed natural language understanding system with default theory,* Proc. of the Seventh International Symposium on Methodologies for Intelligent Systems (ISMIS'93), Trondheim, Norway, June 1993.
6. Honghua Gan: *Planning and understanding with default theories,* Proc. of the Third International Conference for Young Computer Scientists (ICYCS'93), Beijing, P. R. China, July 1993.
7. Honghua Gan: *Default causal reasoning in common sense situations,* PhD thesis, University of Exeter (in preparation), 1994.
8. Judith A. Hudson, Robyn Fivush and Janet Kuebli: *Scripts and Episodes: the Development of Event Memory,* Applied Cognitive Psychology, vol. 6, 483-505 (1992).
9. Yumi Iwasaki and Herbert A. Simon: *Theories of Causal Ordering: Reply to de Kleer and Brown,* Artificial Intelligence 29 (1986) 63-67.
10. J. L. Mackie: *The Cement of the Universe: A Study of Causation,* Oxford University Press, 1974.
11. R. Patil: *Causal representation of patient illness for electrolyte and acid-based diagnosis,* Technical Report 267, Laboratory of Computer Science, MIT (1981).
12. Raymond Reiter: *A Logic for Default Reasoning* Artificial Intelligence 13 1980;
13. Chuck Rieger and Milt Grinberg: *The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms,* IJCAI-77, Vol 1 250-256
14. Roger C. Schank: *Conceptual Dependency: A Theory of Natural Language Understanding,* Cognitive Psychology 3, 552-631 (1972).
15. Roger C. Schank and Robert P. Abelson: *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures,* The Artificial Intelligence Series, Lawrence Erlbaum Associates, Publishers, 1977.
16. Roger C. Schank: *Dynamic Memory: a theory of reminding and learning in computers and people,* Cambridge University Press 1982.
17. Yoav Shoham: *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence,* The MIT Press Series in Artificial Intelligence, The MIT Press, 1988.
18. Herbert A. Simon: *Nonmonotonic Reasoning and Causation: Comments,* Cognitive Science 15, 293-300 (1991).
19. Ernest Sosa: *Causation and Conditionals,* ed., Oxford University Press, 1975.
20. Christopher Nigel Taylor: *A Formal Logical Analysis of Causal Relations,* PhD Thesis, University of Sussex, 1992.

Dealing with Qualitative and Quantitative Temporal Information concerning Periodic Events*

Paolo Terenziani
Dipartimento di Informatica, Universita' di Torino
Corso Svizzera 185, 10149 Torino, Italy

Abstract

The paper describes a framework for representing and reasoning with periodic events. It proposes a temporal formalism, which deals with both (i) quantitative information concerning the frame of time (e.g., *from 1990 until 1993*) and the calendar-dates (e.g., *on Mondays*) in which periodic events take place and (ii) the qualitative relations between periodic events (e.g., *Mary works after John*). The paper defines the basic operations of inversion, intersection and composition of temporal specifications expressed in the given formalism.

1 Introduction

Periodic events are widely studied in many research areas, such as Artificial Intelligence (AI), Temporal Databases (TDB), Concurrent Programs Verifications. In particular, most AI and TDB approaches aim at providing a high-level, powerful and user-friendly formalism for representing periodic events, and (especially in AI) some form of temporal reasoning operating on such formalisms. The work reported in this paper belongs to such a stream of research, aiming at providing a framework in which it is possible to deal also with very rich and complex temporal specifications, as in

Ex.1 *"From 1-1-1992 to 1-6-1993, each Monday Mary worked after John"*

Ex.2 *"Since 1-1-1991 John pays the car assurance on the first Monday of April, before he works"*

(following [Van Eynde, 87], we call *Frame-Time* the interval which contains all the instances of the event -e.g., *"From 1-1-1992 to 1-6-1993"* in Ex.1-, *I-Time* the periodic time interval (calendar-date) over which periodic events take place -e.g.,

* This work was partially supported by the Italian CNR, project "Ambienti e strumenti per la gestione di informazioni temporali".

"Monday" in Ex.1- and *e-Time* the time in which the actual instance of the periodic event occurred -e.g., *"after John -worked-"*).

Current AI and TDB approaches do not allow one to deal with such complex specifications. In particular, these approaches can be roughly divided into two mainstreams, depending on the types of temporal information they deal with. In the first mainstream (carried on especially in the TDB community), most attention has been devoted to the treatment of I-Times and Frame-Times (see, for instance, [Leban et al., 86], [Chomicki & Imielinsky, 88], [Kabanza et al., 90], [Baudinet, 93], [Chandra et al., 93]). For instance, in [Leban et al., 86] it is defined a formalism in which complex specifications of calendar-dates (I-Times) can be built incrementally, on the basis of their composing terms, and in [Chandra et al., 93] it is described an algorithm which, given a Frame Time, generates all of the instances of an I-time expressed in Leban's formalism in the given Frame-Time. In the second mainstream (carried on especially in the AI community) most attention has been paid to the treatment of quantifiers and e-Time, dealing with qualitative relations between periodic events (see, for instance, [Ladkin, 86a, 86b], [Poesio, 88], [Ligozat, 91], [Morris et al., 93]). In particular, in Ladkin's work, the temporal extent of a periodic event is a non-convex time interval, so that a set of basic qualitative relations between non-convex interval is introduced in order to deal with the e-Time. On the other hand, [Morris et al., 93] introduce operators for accessing the convex subintervals composing a non-convex interval, so that Morris can use standard relations between convex (sub)intervals (see [Allen, 83]) for specifying the e-Time and standard reasoning techniques for reasoning with it (see, e.g., [Allen, 83]).

This paper describes an integrated approach, aiming at combining the expressive power of these two complementary mainstreams in order to provide a powerful framework in which complex temporal specifications (see, e.g., Ex.1 and Ex.2) can be represented. Moreover, it also defines the basic operations of inversion, intersection and composition of temporal specifications. These operations are the basis upon which a reasoning mechanism for amalgamating temporal specifications (e.g., for inferring from Ex.1 and Ex.2 that from 1-1-1991 to 1-6-1993 each first Monday of April Mary worked after John paid the car assurance) is built.

The paper is organised as follows. Section 2 introduces the formalism for describing periodic events (and, in particular, their I-times and e-Times). Section 3 proposes a set of basic relations between I-Times. In section 4 these relations are used in order to define the operations of inversion, intersection and composition, which are the core of the reasoning process. Finally, section 5 proposes comparisons and future work.

2 Temporal Representation of Periodic Events

We assume time to be a linear order on a domain consisting of points. A *time interval* I is a convex set of points between a starting (start(I)) and an ending (end(I)) point. As in [Leban et al., 86], we define a *collection* of intervals as a structured and ordered set of non-overlapping intervals. The *depth* (*order* in [Leban et al., 86]) of a collection is the measure of the depth of the structure. *Calendars* are infinite collections of intervals which span the timeline (i.e., such that I_i Meets I_{i+1}, for all i). For example, the collection of days is a calendar. Time intervals are the temporal component of the representation of events. Collections of time intervals are the temporal component of the representation of periodic events, i.e., they represent the collection of the temporal extents upon which the instances (realisations) of the periodic event occur.

The formalism we use for specifying *I-Times* is that in [Leban et al., 86], who introduced a notation for defining basic calendars and two types of operators on collections of intervals (dicing and slicing) for building new (user-defined) collections on the basis of the basic calendars. For example, given the basic definitions of Days*, Weeks* and Months*, the collection of the first Mondays of Aprils can be (incrementally) defined as follows:

(s1) Mondays* ≡ 2 / Days* :during: Weeks*
(s2) Aprils* ≡ 4 / Months* :during: Years*
(s3) First-Monday-of-Aprils* ≡ 1 / Mondays* :during: Aprils*

As shown in Ex.1 and Ex.2, the specification of the *e-Time* of a periodic event ev1* may involve the description of the qualitative temporal relations between ev1* and other periodic events (or, better, between the collections of time intervals in which they happen). In order to deal with such temporal specifications, we adopt the qualitative relations between pair of convex time intervals of Allen's Interval Algebra [Allen, 83]. However, since Allen's relations hold between time intervals, and periodic events happen over collections of time intervals, a way for relating pairs of time intervals belonging to different collections is needed. As in [Morris et al., 93], we introduce the relation of *correlation* (COR) between pairs of instances of periodic events which holds as a result of some contingent relation in the world between them. For instance, in Ex.1 a correlation relation holds between each correlated pair of John's work and Mary's work, and the temporal relation "after" holds between each pair of corresponding time intervals.

However, in our approach, qualitative relations between periodic events are not absolute, but hold in a context (more specifically, in a Frame-Time and an I-Time). Thus, a relation associating (the temporal extents of) the instances of a periodic event to the instances of the I-Time in which they occur must be introduced (this relation is called ASSOC -a shorthand for *association*- in the following). In particular, if n/C is the n-th instance of the collection C, and k/ev1* is the k-th instance of the periodic event ev1*, $ASSOC_{n/C}$(k/ev1*) holds if and only if k/ev1* happened in n/C.

Finally, we face co-designation problems [Morris et al., 93] by indexing the description of periodic events with constant symbols. For instance, $John\text{-}wash\text{-}teeth_1$ and $John\text{-}wash\text{-}teeth_2$ indicate the two different sequences of periodic events of the same type in Ex.3

Ex.3 *"Each day, John washes his teeth after breakfast and before going to sleep"*

Therefore, in our approach, complex specifications of periodic events can be provided, according to the following schema[1]:

[1] For the moment, we do not cope with "generalised" temporal quantifiers, such as "never", "sometimes", "only", "always and only" in [Morris et al., 93], "partially", "mostly" etc. in [Ladkin, 86b]. For the sake of brevity, in this paper we do not deal with cases in which the Frame Time and/or the I-Time are omitted, and with purely quantitative information as in "[1-1-1992, 1-6-193] John-works* EACH Mondays".

<Frame> e-id$_k$* <Quant> <I-Time> <Qual-Rel> e-id$_h$*

where <Frame> is specified as the range of time spanning between a starting point and an ending point (e.g. [1-1-1992, 1-6-1993]), <Quant> is the quantifier "EACH", <I-Time> is specified as in [Leban et al., 86] and <Qual-Rel> is a (possibly ambiguous) relation in Allen's Interval Algebra. The meaning of a temporal specification is the following: in [d1, d2] the periodic events ev1$_k$* and ev2$_h$* occur exactly once in each instance of C, and the relations between each correlated pair of occurrences is R. More formally:

[d1, d2] ev1$_k$* EACH C R ev2$_h$* iff for each instance n/C of C in the Frame Time [d1, d2], there is exactly one instance j/ev1$_k$* of ev1$_k$* and one instance i/ev2$_h$* of ev2$_h$* and the following holds: ASSOC$_{n/C}$(j/ev1$_k$*) and ASSOC$_{n/C}$(i/ev2$_h$*) and COR(j/ev1$_k$* , i/ev2$_h$*) and j/ev1$_k$* R i/ev2$_h$*.

The expressive power of this formalism is very high. For instance, the temporal content of Ex.1 and Ex.2 above can be represented as follows (given the definitions of I-Times in (s1-s3) and supposing that Ex.1 and Ex.2 co-designate the same periodic event of John's work):

(s4) [1-1-1992, 1-6-1993] Mary-works$_k$* EACH Mondays* (AFTER) John-works$_j$*

(s5) [1-1-1991 , now] John-pays-car-assurance$_i$* EACH First-Monday-of-Aprils* (BEFORE) John-works$_j$*

3 Relations between I-Times

In order to define the basic operations (inversion, intersection and composition) between temporal specifications of periodic events, the possible relations between I-Times must be analysed. Four basic binary relations between I-Times are relevant to this purpose:

- Given two I-Times c1* and c2*, **c1* $\prec$ c2*** (read as: c1* is more specific that c2*) iff for each instance of c1* there is exactly one instance of c2* which properly contains it and, conversely, for each instance of c2* there is exactly one instance of c1* which is contained in it (1:1 correspondence) (e.g., Mondays* $\prec$ Weeks*). More formally: c1* $\prec$ c2* iff $\forall$i $\exists$! j such that i/c1* (STARTS, DURING, FINISHES) j/c2* and $\forall$j $\exists$!i such that j/c2* (STARTED-BY, CONTAINS, FINISHED-BY) i/c1*
- Given two I-Times c1* and c2*, **c1* $\in$ c2*** (read as: c1* is a restriction of c2*) iff for each instance of c1* there is an instance of c2* which is temporally equal to it, and there is a correspondence 1:n between instances of c1* and instances of c2* (e.g., Mondays* $\in$ Days*). More formally, c1* $\in$ c2* iff $\forall$i $\exists$! j such that i/c1* (EQUAL) j/c2* and NOT ($\forall$j $\exists$i such that j/c2* (EQUAL) i/c1*)
- Given two I-Times c1* and c2*, **c1* $\subset$ c2*** (c1* is contained into c2*) iff there are at least two instances of c1* which are contained into the same instance of c2* (e.g., Days* $\subset$ Weeks*, Mondays* $\subset$ Aprils*). More formally, c1* $\subset$ c2* iff $\exists$

i,j,k such that i/c1* (STARTS, DURING, FINISHES) k/c2* and j/c1* (STARTS, DURING, FINISHES) k/c2*

The relations $\prec$, $\in$ and $\subset$ are antisimmetric and transitive.

- **c1* = c2*** , defined in the obvious way.

For the sake of convenience, we also introduce the relation **c1* # c2*** (read as: c1* and c2* are temporally incomparable), holding between c1* and c2* iff none of the above relations hold between them.

In some cases, given two I-Times c1* and c2*, the relations between them can be evaluated on the basis of their definitions. For instance, the definition c1* $\equiv$ n / c2* :during: c3* implies that c1* $\prec$ c3* and c1* $\in$ c2* (e.g., from Aprils* $\equiv$ 4 / Months*:during:Years* we have that Months* $\subset$ Years* and Aprils* $\prec$ Years* and Aprils* $\in$ Months*). However, in case two I-Times are defined by independent specifications, it is sometimes difficult to determine the relations between them only on the basis of their specifications, since the user is completely free in the use of the specification language for I-Times (see [Leban et al., 86]). Currently, in these cases the relations are directly asked to the user.

4 Basic Operations on Periodic Events Specifications

The basic operations of inversion (indicated as $^{-1}$), check-intersection ($\underline{\cap}$) and composition (@) of periodic events specifications can now be defined. Since the specification of periodic events is structured, these operations have to operate on the different parts of the specification, namely the Frame Times, the I-Times and the qualitative temporal relations. As regards the operations on the qualitative temporal relations, we import the inversion, intersection and composition operations of Allen's algebra [Allen, 83] (indicated as $^{-1}$e, $\cap_e$ and $@_e$ respectively; "e" indicates that we apply these operators to the e-Time of the temporal specifications).

4.1 Inversion

Given a temporal specification [d1, d2] $ev1_i$* Q c1 R1 $ev2_j$*, stating the relations between two periodic events $ev1_i$* and $ev2_j$*, its inverse (indicated as ([d1, d2] $ev1_i$* Q c1 R1 $ev2_j$*) $^{-1}$) establishes the corresponding relations between $ev2_j$* and $ev1_i$*. The inversion operation is based on the inversion in Allen's Interval Algebra: the inverse specification is simply obtained by inverting the relations between the e-Times:

([d1, d2] $ev1_i$* Q C1 R1 $ev2_j$*) $^{-1}$ = [d1, d2] $ev2_j$* Q C1 R1^{-1}e $ev1_i$*

e.g.: ([1-1-92, 15-2-92] a* EACH Days* (BEFORE,MEETS) b*) $^{-1}$ =
[1-1-92, 15-2-92] b* EACH Days* (AFTER,MET-BY) a*
(here and in the following a*, b* and c* represent periodic events).

4.2 Check-Intersection

In our approach, it is possible to specify different temporal relations between the same pair of periodic events, holding at different Frame Times. Even in the same Frame Time, different specifications may be allowed (consider, for instance (s6) and (s7) below). However, the consistency of the temporal specifications on the "overlapping parts" of the Frames Times and of the I-Times must be checked. For instance, given (s6), the AFTER relation between a* and b* asserted in (s7) is not possible since 1-1-91 until 1-1-92, and must be ruled out.

(s6) [1-1-91, 1-1-93] a* EACH Days* (BEFORE,MEETS) b*
(s7) [1-1-91, 1-1-92] a* EACH Mondays* (BEFORE,AFTER) b*

Thus, since in our framework temporal relations are "context-dependent", it seems not useful to define a "standard" intersection operation, providing the unique temporal constraint between a pair of temporal entities, as for instance in [Allen, 83] (we call our "non-standard" operation "check-intersection" -indicated as $\underline{\cap}$-). Check-intersection operates on two temporal specifications involving the same pair of periodic events (of the type of (s8) and (s9)) and works in two steps:

(s8) [d1', d1"] $ev1_i$* Q C1 R1 $ev2_j$*
(s9) [d2', d2"] $ev1_i$* Q C2 R2 $ev2_j$*

First, the intersection of the two Frame Times is computed. If it is empty, then no further operation must be devised, and the original specifications are left unchanged. Otherwise,

(i) the original specifications are left unchanged as regards the non-intersecting parts of the Frame Times
(ii) as regards the intersecting part of the Frame Times, check-intersection checks whether there is an "intersection" also between the I-Times and, in this case, the compatibility of the temporal relations is forced.

More specifically, the value of the check-intersection operation

(s10) [d1', d1"] $ev1_i$* Q C1 R1 $ev2_j$* $\underline{\cap}$ [d2', d2"] $ev1_i$* Q C2 R2 $ev2_j$*

in the intersection [d1', d1"] ∩ [d2', d2"] of the two Frame Times is

(s11) $ev1_i$* Q C1 R1 $ev2_j$* $\underline{\cap}'$ $ev1_i$* Q C2 R2 $ev2_j$*

where $\underline{\cap}'$ indicates the second part of check-intersection (called check-intersection'), operating on two partial temporal specifications (in the sense that the Frame Time in no longer considered). In the definition of $\underline{\cap}'$ different cases must be distinguished, depending on the relations between the related I-Times. In the following, R indicates the intersection between the Interval Algebra relations R1 and R2 (i.e., R = R1 $\cap_e$ R2).

- **C1 = C2**

If R is the null relation, an inconsistency is reported. Otherwise, the result of check-intersection' is $ev1_i$* Q C1 R $ev2_j$*. For example:
a* EACH Days* (BEFORE,MEETS) b* $\underline{\cap}'$
a* EACH Days* (BEFORE,OVERLAPS) b* --> a* EACH Days* (BEFORE) b*

- **C1 $\subset$ C2 (the same holds if C2 $\subset$ C1)**

In such a case an inconsistency is reported. For example:
a* EACH Days* (BEFORE,MEETS) b* $\underline{\cap}'$ a* EACH Months* (BEFORE) b*
gives an inconsistency, since a* cannot happen exactly once a day and exactly once a month.

- **C1 $\prec$ C2 (the inverse holds if C2 $\prec$ C1)**

If R is empty, an inconsistency is reported. Otherwise, the result of check-intersection' is ev1$_i$* Q C1 R ev2$_j$* (the most specific qualitative temporal relations and I-Times are selected). E.g.,
a* EACH Mondays* (BEFORE) b* $\underline{\cap}'$ a* EACH Weeks* (BEFORE,MEETS) b*
--> a* EACH Mondays* (BEFORE) b*

- **C1 $\in$ C2 (the inverse holds if C2 $\in$ C1)**

If R is empty, an inconsistency is reported. Otherwise, the result of check-intersection' is ev1$_i$* Q C1 R ev2$_j$* and ev1$_i$* Q C2 R2 ev2$_j$*. In fact, the qualitative relations holding in the restricted I-Time must be forced to be compatible those holding in general. For example:
a* EACH Mondays* (BEFORE,AFTER) b* $\underline{\cap}'$
a* EACH Days* (BEFORE,OVERLAPS) b* -->
a* EACH Mondays* (BEFORE) b* and a* EACH Days* (BEFORE,OVERLAPS) b*

- **C1 # C2**

The temporal specifications provide two different constraints between the same pair of periodic events, holding at "incomparable" I-times. In such a case no new qualitative constraint can be inferred, and the input constraints are left unchanged (with the implicit meaning that both of them must hold in the intersection -if any- of the instances of the two I-Times). For example:
a* EACH Mondays* (MEETS) b* $\underline{\cap}'$ a* EACH Tuesdays* (AFTER) b* --> UNCHANGED

4.3 Composition

Composition (@) applies to two different specifications of periodic events which involve the same periodic event (i.e., which co-designate the same periodic event; e.g., ev1$_i$* in (s15))

(s12) [d1', d1"] ev1$_i$* Q C1 R1 ev2$_j$* @ [d2', d2"] ev1$_i$* Q C2 R2 ev3$_k$*

As in the case of check-intersection, no new information can be inferred as regards the non-intersecting parts of the two Frame Times. However, in the time interval [d1', d1"] ∩ [d2', d2"] (if any), not only is it possible to infer the temporal constraints between ev2$_j$* and ev3$_k$*, but, in some cases, it is also possible to detect an inconsistency in the temporal specifications of the I-Times associated to ev1$_i$*. In fact, [d1', d1"] ev1$_i$* Q C1 R1 ev2$_j$* and [d2', d2"] ev1$_i$* Q C2 R2 ev3$_k$* also mean that, among the other things, (in [d1', d1"]) ev1$_i$* happens exactly once in each occurrence of C1, and (in [d1', d2"]) ev1$_i$* happens exactly once in each occurrence of C2. Thus, the compatibility of these two constraints has to be checked, during the composition. Then, the result of the composition operation

(s13) ev1$_i$* Q C1 R1 ev2$_j$* @' ev1$_i$* Q C2 R2 ev3$_k$*

(where @' indicates the operation of composition restricted to consider all the parts of the temporal specification except the Frame Time) in [d1', d1"] $\cap$ [d2', d2"] is an inconsistency or a specification of the type

(s14) $ev2_j$* Q C R $ev3_k$*

where R represents the composition, in Allen's algebra, of the inverse of R1 with R2 ($R = R1^{-1}e\ @_e\ R2$) and C is the I-Time resulting from the composition. The definition of @' (composition') depends on the relations between the related I-Times C1 and C2. Different cases can be distinguished.

- **C1 = C2**

The final composition' of the two constraints is $ev2_j$* Q C2 R $ev3_k$*. For example:
a* EACH Days* (BEFORE,MEETS) b* @ a* EACH Days* (AFTER) c* -->
b* EACH Days* (AFTER) c*

- **C1 $\subset$ C2 (the same holds if C2 $\subset$ C1)**

An inconsistency is reported, since it cannot be the case that $ev1_i$* happens exactly once both in C1 and in C2. For example:
a* EACH Days* (BEFORE) b* @ a* EACH Months* (AFTER) c* --> INCONSISTENT

- **C1 $\prec$ C2 (the inverse holds if C2 $\prec$ C1)**

The result of the composition is $ev2_j$* Q C2 R $ev3_k$* [2]. For example:
a* EACH Tuesdays* (BEFORE,OVERLAPS) b* @ a* EACH Weeks* (AFTER) c* --> b* EACH Weeks* (AFTER) c*

- **C1 $\in$ C2 (the inverse holds if C2 $\in$ C1)**

The result of the composition is $ev2_j$* Q C1 R $ev3_k$*. Notice that C1 is the I-Time for the new constraints between $ev2_j$* and $ev3_k$*. For example:
a* EACH Mondays* (BEFORE,MEETS,OVERLAPS) b* @
a* EACH Days* (AFTER) c* --> b* EACH Mondays* (AFTER) c*

- **C1 # C2 (C1 and C2 are not comparable)**

No new qualitative relation can be inferred. For example:
a* EACH Mondays* (BEFORE,MEETS) b* @ a* EACH Tuesdays* (AFTER) c*
--> NO NEW TEMPORAL CONSTRAINT

4.4 Reasoning Process

Intersection and composition are used in an Allen's like path-consistency constraint propagation algorithm [Allen, 83] in order to amalgamate a K.B. of temporal specifications of periodic events. In particular, for each triple of periodic events a*, b*, c* in the K.B., the temporal constraint between a* and c* is computed by composing (via @) the temporal constraints holding between a* and b* with those holding between b* and c*. Then, the result of the composition is intersected with each one of the constraints (holding at different I-times and Frame Times) between a* and c*. The process is iterated until a state of quiescence is reached.

[2] In fact, it would not be correct to infer that $ev3_k$* happens exactly once each C1. For instance, in the example below, it is only implied that c* happens once a week before b* (which is on Tuesdays), but it is not possible to infer the exact day in which it happens.

5 Comparisons and Developments

The paper describes a new approach to the treatment of periodic events dealing with both quantitative information concerning the Frame-Time and the I-Time in which periodic events are located and with the qualitative relations between periodic events (e-Time).

The proposed approach compares favourably with the other approaches dealing with periodic events in the AI and in the TDB literature. In fact, it proposes an integration of the works focusing only on the specification of calendar-dates (I-Times, in our terminology; see, for instance, [Leban et al., 86], [Chomicki & Imielinsky, 88], [Kabanza et al., 90], [Baudinet, 93], [Chandra et al., 93]) and those dealing with the qualitative relations between periodic events (e-Times, in our terminology; see, for instance, [Ladkin, 86a, 86b], [Poesio, 88], [Ligozat, 91], [Morris et al., 93]). In particular, our approach is different from all the other approaches dealing with qualitative temporal relations between periodic events in the literature (to the best of our knowledge), since, in our proposal, qualitative relations hold in a context (more specifically, in a Frame-Time and an I-Time). In our opinion this constitutes a fundamental improvement over previous works. In fact, most qualitative relations between periodic events in the world are context-dependent. Thus, the practical applicability of approaches such as those in [Ladkin, 86a, 86b], [Ligozat, 91], [Morris et al., 93] is very limited, since they can only express context-independent ("absolute") qualitative constraints between periodic events (e.g. *Mary always/sometimes works after John")*. In particular, we plan to apply our approach to scheduling and office automation problems, where the treatment of context-dependent temporal constraints between periodic events is crucial (for instance, it is important to be able to state temporal constraints such as "the activity A must precede B on Mondays, and follow B on Tuesdays").

Nevertheless, many other aspects have to be taken into account in order to obtain a comprehensive approach to periodic events. In particular, we are currently investigating the possibility of extending or framework for dealing with quantifiers such as "only", "sometimes" etc. in [Morris et al., 93]. However, the interplay between I-Times and quantifiers have to be carefully considered. Moreover, we are analysing the possibility of considering also quantitative temporal information in the specifications of e-Times, such as temporal locations and durations. Finally, we are also analysing whether, with suitable restrictions on the specification language for I-Times, it is possible to obtain a complete algorithm for determining the relations between I-Times on the basis only of the parser tree (determined as in [Chandra et al., 93]) of the specifications.

Acknowledgements

I thank Prof. Pietro Torasso of the Dipartimento di Informatica of the Universita' di Torino for his constant cooperation and his useful advice.

References

[Allen, 83] J.F. Allen: "Maintaining Knowledge about Temporal Intervals", *Comm ACM* 26(11) 832-843 (1983).

[Baudinet et al., 93] M. Baudinet, J. Chomicki, P. Wolper: "Temporal Deductive Databases", in A. Tansell, R. Snodgrass, J. Clifford, S. Gadia, A. Segev (eds): *Temporal Databases: Theory, Design, and Implementation*, Benjamin-Cummings, 1993 (in press)

[Chandra et al., 93] R. Chandra, A. Segev, M. Stonebraker: "Implementing Calendars and Temporal Rules in Next Generation Databases", LBL Tech. Report 34229, University of California, Berkeley (1993).

[Chomicki & Imielinsky, 88] J. Chomicki, T. Imielinsky: "Temporal Deductive Databases and Infinite Objects", *Proc. 7th ACM Symposium on Principles of Database Systems*, 61-73 (1988).

[Kabanza et al., 90] F. Kabanza, J.-M. Stevenne, P. Wolper: "Handling Infinite Temporal Data", *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 392-403 (1990).

[Koomen, 91] J. Koomen: "Reasoning about Recurrence", *International Journal of Intelligent Systems 6*, 461-496 (1991).

[Ladkin, 86a] P. Ladkin: "Primitive and Units for Time Specification", *Proc. AAAI'86*, 354-359 (1986).

[Ladkin, 86b] P. Ladkin: "Time Representation: A Taxonomy of Interval Relations", *Proc. AAAI'86*, 360-366 (1986).

[Leban et al., 86] B. Leban, D.D. McDonald, D.R. Forster: "A representation for collections of temporal intervals", *Proc. AAAI'86*, 367-371 (1986).

[Ligozat 91] G. Ligozat: "On Generalized Interval Calculi", *Proc. AAAI'91*, 234-240 (1991).

[Morris et al., 93] R.A. Morris, W.D. Shoaff, L. Khatib: "Path Consistency in a Network of Non-convex Intervals", *Proc. IJCAI'93*, 655-660 (1993).

[Niezette & Stevenne, 92] M. Niezette, J.-M. Stevenne: "An Efficient Symbolic Representation of Periodic Time", Proc. First International Conference on Information and Knowledge Management, 1992.

[Poesio, 88] M. Poesio: "Toward a Hybrid Representation of Time", *Proc. ECAI'88*, 247-252 (1988).

[Van Eynde, 87] F. Van Eynde: "Iteration, Habituality and Verb Form Semantics", *Proc. 3rd Conference of the European Chapter of the Association for Computational Linguistics*, 270-277 (1987).

Distributed Multi-agent Probabilistic Reasoning With Bayesian Networks

Yang Xiang

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2, yxiang@cs.uregina.ca

Abstract. Main stream approaches in distributed artificial intelligence (DAI) are essentially logic-based. Little has been reported to explore probabilistic approach in DAI. On the other hand, Bayesian networks have been applied to many AI tasks that require reasoning under uncertainty. However, as commonly applied, a single-agent paradigm is assumed in Bayesian networks.

This paper extends multiply sectioned Bayesian networks (MSBNs) for single-agent systems into a framework for multi-agent distributed interpretation systems. Each cooperative agent is represented as a Bayesian subnet that consumes its own computational resource, gathers its own evidence, and can answer queries. Unlike in single-agent systems where evidence is entered one subnet at a time, multiple agents may acquire evidence in parallel. We add to the set of single-agent MSBN operations new belief propagation operations which, when performed, regain global consistency. Due to the inter-agent 'distance' and the associated communication cost, global consistency can not be maintained constantly. We show that, when the proposed operations are followed, between two successive communications, the answers to queries from an agent are consistent with all local evidence gathered so far, and are consistent with all global evidence gathered up to the last communication.

Keywords: knowledge representation and integration, approximate reasoning, probabilistic reasoning, Bayesian networks, distributed artificial intelligence, distributed reasoning.

1 Introduction

Probabilistic reasoning in Bayesian networks (BNs), as commonly applied, assumes a single-agent paradigm: A single processor accesses a single global network representation, updates the joint probability distribution over the network variables as evidence becomes available, and answers queries. Concurrency, as applied to BNs, primarily aims at performance and decentralization of control [10, 4], but not at modeling inference among multiple agents with multiple perspectives. The resultant individual concurrent element is thus 'fine-grained', e.g., a node in a BN [10] or a clique in the junction tree representation of a BN [4].

The single-agent paradigm is inadequate when uncertain reasoning is performed by elements of a system between which there is some 'distance', which may be spatial, temporal, or semantic (elements are specialized differently) [1].

Such systems pose special issues that must be addressed. A multi-agent view is thus required where each agent is an autonomous intelligent subsystem. Each agent holds its own partial domain knowledge, accesses some information source, and consumes some computational resource. Each agent communicates with other agents to achieve the system's goal cooperatively.

Distributed artificial intelligence (DAI), a subfield of artificial intelligence, addresses the problems of designing and analyzing such 'large-grained' coordinating multi-agent systems [2, 3]. Main stream approaches, e.g., blackboard systems, contract nets, and open systems are essentially logic-based. To our best knowledge, little has been reported to explore probabilistic approach in DAI.

This paper reports our pilot study to apply probabilistic approach to distributed multi-agent reasoning. Our representation is based on multiply sectioned Bayesian networks (MSBNs)[12], which were developed for single-agent-oriented, modular knowledge representation and more efficient inference[11]. We show that the modularity of MSBNs allows a natural extension into a multi-agent reasoning formalism.

We address the use of MSBNs in distributed interpretation, a subclass of problems in DAI. As defined originally by Lesser and Erman [7], an *interpretation* system accepts evidence from some environment and produces higher level descriptions of objects and events in the environment. A *distributed* interpretation system is needed when sensors for collecting evidence are distributed, and communication of all evidence to a centralized site is undesirable. Examples of such systems include sensor networks, medical diagnosis by multiple specialists, trouble-shotting of complex artifacts and distributed image interpretation.

Section 2 briefly introduce BNs and single-agent MSBNs. Section 3 presents the semantic extension of single-agent MSBNs to multi-agent MSBNs. Each cooperative agent is represented as a Bayesian subnet that consumes its own computational resource, gathers its own evidence, and can answer queries. Section 4 discusses new technical issues raised due to the extension, which are addressed in subsequent sections.

Section 5 extends the evidence entering operation proposed originally [12] to include both specific evidence and virtual evidence.

Unlike in single-agent systems where evidence is entered one subnet at a time, multiple agents may acquire evidence asynchronously in parallel. Section 6 addresses this and adds new belief propagation operations to the set of single-agent MSBN operations.

Section 7 shows that global consistency is guaranteed when the proposed operations are followed. Due to the inter-agent 'distance' and the associated communication cost, global consistency can not be maintained constantly. It is also shown that, when the proposed operations are followed, between two successive communications, the answers to queries from an agent are consistent with all local evidence gathered so far, and are consistent with all global evidence gathered up to the last communication.

Section 8 discusses the role of causal independence in our framework.

2 Single Agent Oriented MSBNs

A BN [10, 8, 6, 4] is a triplet (N, E, P). N is a set of nodes (variables). E is a set of arcs such that $D = (N, E)$ is a directed acyclic graph (DAG). For each node $A_i \in N$, the strengths of the causal influences from its parent nodes π_i are quantified by a conditional probability distribution $p(A_i|\pi_i)$. The basic dependency assumption embedded in BNs is that a variable is independent of its non-descendants given its parents. P is a joint probability distribution (JPD). For a BN with α nodes, P can be specified by the following product due to the assumption: $P = p(A_1 \ldots A_\alpha) = \prod_{i=1}^{\alpha} p(A_i|\pi_i)$.

In the following, we briefly introduce single-agent oriented MSBNs. For a formal presentation of MSBNs, see [12]. A MSBN consists of a set of interrelated Bayesian subnets. Each subnet represents one subdomain in a large domain. Each subnet shares a non-empty set of variables with at least one other subnet. The interfacing set between each pair of subnets must satisfy a *d-sepset* condition such that, when the pair is isolated from the MSBN, the interfacing set renders the two subnets conditionally independent.

The overall structure with which the subnets of a MSBN are organized is subject to a constraint called *soundness of sectioning.* Without the condition, a MSBN is subject to loss of information when later transformed into its secondary representation: a linked junction forest. A sufficient condition for sound sectioning is to construct a MSBN with a *hypertree* structure. The hypernodes in the hypertree are subnets of the MSBN. The hyperlinks are interfacing sets between subnets. Conceptually, the hypertree structured MSBN is built by adding one subnet (hypernode) to existing ones at a time. Only one interfacing set (hyperlink) to an existing subnet is explicitly stored. A hypertree structured MSBN guarantees that each subnet renders the rest of the MSBN conditionally independent. Figure 1 depicts a hypertree structured MSBN. Each box represents a subnet. Boundaries between boxes represent interfacing sets. The superscripts of subDAGs indicate a possible order of construction.

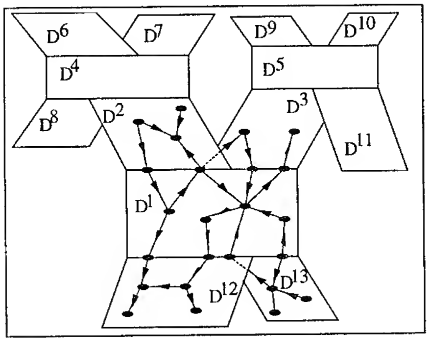


Fig. 1. A MSBN with a hypertree structure.

Once a hypertree structured MSBN is constructed, it is then converted into a *linked junction forest* of the identical hypertree structure. Each hypernode in the hypertree is a *junction tree* (clique tree, join tree) converted from a subnet in the hypertree structured MSBN. The conversion of a subnet to a junction tree is subject to a *separability* constraint to guarantee non-distorted belief propagation. Each hyperlink of the hypertree is a set of *linkages* converted from the corresponding interfacing set in the hypertree structured MSBN.

Parallel to the transformation of the graphical structure, there is a corresponding transformation of the probability tables in the MSBN to *belief tables* in the junction forest. The conversion is subject to a *supportiveness* constraint such that belief can be propagated without blocking. The overall conversion guarantees a *joint system belief* of the linked junction forest can be assembled from belief tables distributed in the forest, which is equivalent to the joint probability distribution of the MSBN.

To answer queries by efficient local computation in a linked junction forest (LJF), the LJF must be made consistent. A LJF is *Locally consistent* if all junction trees (JTs) are internally consistent, i.e., when marginalized onto the same set of variables different belief tables in a junction tree yield the same marginal distribution. A LJF is *boundary consistent* if each pair of linked JTs are consistent with respect to their interfacing set. A LJF is *globally consistent* iff it is both locally consistent and boundary consistent.

A set of operations are developed to achieve consistency in a LJF during evidential reasoning: **BeliefInitialization** establishes initial global consistency. **DistributeEvidence** causes an outward belief propagation within a JT, and brings the JT internally consistent after evidence on variables in a single clique has been entered. **CollectEvidence** causes an inward belief propagation within a JT. **UnifyBelief** brings a JT internally consistent after evidence on variables in multiple cliques has been entered. **EnterEvidence** updates belief in a JT in light of new evidence, and brings the JT internally consistent again by calling either **DistributeEvidence** or **UnifyBelief**. **UpdateBelief** updates the belief of a JT T relative to an adjacent JT, and brings T internally consistent. **DistributeBelief** initiated at a JT T causes an outward belief propagation in the LJF radiating from T. **ShiftAttention** allows the user to enter multiple pieces of evidence into a JT of current attention, and, when the user shifts attention to a target JT, maintains consistency along the hyperpath in the hypertree structured forest from the current JT to the target.

3 Representing Multiple Agents in MSBNs

A MSBN represents a large domain by representing a set of subdomains. From the viewpoint of reasoning agents, a MSBN represents multiple perspectives of a single agent. For example, PAINULIM [11] consists of three subnets which represents a neurologist's three different perspectives of the neuromuscular diagnostic domain: clinical, EMG and nerve conduction perspectives.

In a multi-agent system, each agent can be considered as holding its own perspective of the domain. This partial perspective may be over a subdomain, a period of time, or a geographical area. The modular representation of MSBN thus allows a natural extension to multi-agent system, with a modification of the semantics: Instead of representing *one* agent's *multiple* perspectives of a domain, a multi-agent MSBN represents *multiple* agents in a domain each of which holds *one* perspective of the domain. Each subnet corresponds to one such perspective. We extend the example by Lauritzen and Spiegelhalter [6] to illustrate this:

Example 1 Dyspnoea (δ) may be due to tuberculosis (τ), lung cancer (ι) or bronchitis (β). A recent visit to Asia (ν) increases the chances of τ, while smoking (ζ) is a known risk factor for both τ and ι. After an initial diagnosis based on the above information, to further discriminate between τ and ι, a clinician may request lab tests from a radiology lab and a biology lab. Radiology lab has two relevant tests for τ and ι: X-ray (χ) and laminagraphy (α). Biology lab has two relevant tests as well: sputum test (ρ) and biopsy (o). Lab reports describe their impression upon τ and ι based on the results of the test(s).

The above fictitious example involves three agents: a clinical doctor, a radiologist, and a biologist. Diagnosis of a patient with dyspnoea is their common goal. But each agent has its own perspective of the same patient. The distance between them is semantic.

A multi-agent MSBN can be constructed as a decision aid in such a domain. The MSBN consists of three agents: clinical, radiological, and biological subnets. The three agents may process evidence in parallel. Though it makes sense that the radiologist and biologist may not be involved until the clinician's initial diagnosis has been reached, and the patient has to visit them in sequence, considering the time delay required to develop lab results, the radiological and biological subnets may well receive evidence and be posed with queries at the same time. The resultant multi-agent MSBN is illustrated in Figure 2. The transformed LJF is shown in Figure 3.

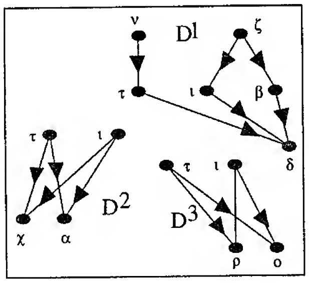


Fig. 2. A multi-agent MSBN representing three medical specialists diagnosing a patient with dyspnoea. D^1: clinical subnet, D^2: radiological subnet, D^3: biological subnet.

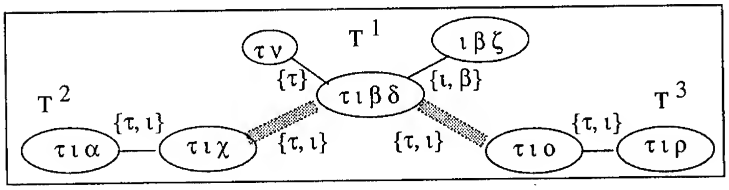


Fig. 3. The linked junction forest for the multi-agent MSBN in Figure 1. Sepsets between cliques are shown in solid lines. Linkages are shown in dotted lines.

4 Consistency Issues in Multi-agent MSBNs

The natural extension of MSBNs to multi-agent systems implies that all the technical constraints applicable to the construction of single-agent MSBNs must be followed in the construction of multi-agent MSBNs. In addition, evidential reasoning in multi-agent systems raises new issues regarding consistency, which must be addressed. To appreciate the issues, we first review how consistency is maintained in single-agent MSBNs:

Initial global consistency is obtained by **BeliefInitialization** (Section 2). The agent focuses attention on one subdomain at a time. Therefore, evidence is entered *one subnet at a time*. As the agent shifts attention to each subnet and enters evidence, **ShiftAttention** (Section 2) maintains (local) consistency along the hyperpath in the hypertree structured forest. It is proven [12] that such local consistency is actually at the global level, i.e., answers to queries at the current JT is consistent with the evidence acquired in the entire LJF.

4.1 How to regain global consistency?

Notice that **ShiftAttention** being able to maintain local consistency at the global level depends directly on the fact that evidence is always entered at the current subnet and nowhere else. In a multi-agent system, multiple agents may acquire evidence asynchronously in parallel. **ShiftAttention** can no longer be used to maintain global consistency. It must be replaced by new operations which we shall refer to as *communication*. We propose the communication operations in Section 6 and prove their property in Section 7.

4.2 What is the consistency level between communications?

ShiftAttention maintains directly only local consistency. But the local consistency happens to be at the global level in a single-agent MSBN! In a multi-agent MSBN, local consistency at a JT means one thing, and the global consistency means another. We can no longer shot two birds with one stone.

Even with new communication operations, due to the inter-agent 'distance' and the associated communication cost, global consistency can not be maintained constantly. A question that must be answered is, between two successive communications which regain global consistency, what is the consistency level of each agent after acquiring additional evidence? We prove a theorem to answer this question in Section 7.

5 Entering Virtual Evidence

Entering virtual evidence (described below) to a directed tree has been discussed in [8]. Undirected tree representations (junction tree or LJF) provide more flexible inference mechanism. But to our best knowledge, virtual evidence has not been treated formally in such representations. Evidence entering operations for such representations only considered entering specific evidence (described below) into a junction tree [5] or a LJF [12]. To be complete, in this section, we extend **EnterEvidence** to include entering virtual evidence into a LJF. This result is applicable to both single-agent MSBNs and multi-agent MSBNs. Our presentation assumes some familiarity with the technical details of MSBN construction. Readers who are interested in results pertinent to only multi-agent reasoning can skip this section, and assume the previous **EnterEvidence** operation for the rest of the paper.

A *piece* of evidence is obtained by one agent's observation, made at one time, of a set of variables contained in its subnet. Different agents may acquire pieces of evidence from their local sources asynchronously in parallel.

Pearl [10] classifies evidence into *specific* and *virtual.* Specific evidence on a variable is obtained by a direct observation of the value of the variable. Virtual evidence "corresponds to judgments based on undisclosed observations that are outside the network but have a bearing on variables in the network". For example, an X-ray image of chest may be regarded as normal or abnormal. Since there is no universally agreeable way to determine mechanically the status of an X-ray image, a radiologist must interpret the pattern subjectively.

Each piece of evidence is represented by an evidence function. See [12] on the evidence function for specific evidence. We first consider a piece of virtual evidence that involves only a single variable. Suppose the virtual evidence has a direct bearing on a variable A. Pearl [10] suggests to code such evidence into a BN by adding a child node e to A and assessing the distribution $p(e|A)$. Following this representation, we analyze its impact on the LJF. Since e is contained in only one subnet in the MSBN, it is sufficient to analyze its impact on the subnet/JT that contains e. We shall denote them by S and T, respectively.

During moralization and triangulation, since the node e is a leaf with a single parent, no link is added between e and other nodes in S. Hence the only clique that e will participate is $C_0 = \{e, A\}$. During construction of T, since the only possible variable that C_0 can share with other cliques is A, C_0 can always be configured as a leaf in T with an arbitrary neighbour clique C such that $A \in C$.

Suppose T is internally consistent before e is observed. After e is observed, *DistributeEvidence* can be called on C_0. The updated belief table (BT) on C is $B'(C) = B(C) * B'(A)/B(A)$ where $B(A)$ and $B'(A)$ stand for the previous and the updated sepset BTs, respectively. Since e is observed as $True$, $B'(A) = \sum_e B'(C_0) = \sum_e B'(eA) = B'(eA) = p(e|A) * B(A)$. Substituting $B'(A)$ in the above equation, we have $B'(C) = B(C) * p(e|A)$. This means that virtual evidence can be represented and entered in a LJF in exactly the same way as specific evidence, and we do not need to create the clique C_0 at all in the first place. We summarize the above analysis in the following definition:

Definition 2 (Evidence function) *Let $S = (N, E, P)$ be a subnet in a MSBN. Let $X \subset N$ be a set of variables involved in a piece of evidence, and let the space of $A \in X$ be $\mathcal{A}$. An evidence function $f : \bigcup_{A \in X} \mathcal{A} \rightarrow [0, 1]$ is defined by the following rule:*

1. *If A is directly observable, for each $a \in \mathcal{A}$ assign to $f(a)$ either 1 or 0 depending on whether a is still a possible outcome of A.*

2. *If A's value is not directly observable, but is interpreted by an autonomous interpreter based on an observation e, then assign to $f(a)$ the subjective conditional probability $p(e|a)$ of the interpreter.*

Let $f : X \rightarrow Z$ be a function. Let $Y \subset X$ be a subset of the domain X. We define the *restriction* of f restricted to Y as a function $f_Y : Y \rightarrow Z$ according to the rule: for each $y \in Y$, $f_Y(y) = f(y)$.

The extended **EnterEvidence** operation that handles both specific and virtual evidence is defined as follows:

Operation 3 (EnterEvidence) *Let T be a JT in a LJF. Let X be a set of variables in T that is involved in a piece of evidence E represented by an evidence function $f : \bigcup_{A \in X} \mathcal{A} \rightarrow [0,1]$. When* **EnterEvidence** *is initiated at T to enter E, the following are performed: (1) For each $A \in X$, a belief universe $U = (C, B(C))$ such that $A \in C$ is arbitrarily selected, and $B(C)$ is multiplied by f restricted to $\mathcal{A}$. (2) If $X \subseteq C$, i.e., only a single universe is involved in the above step,* **DistributeEvidence** *is called in U, otherwise* **UnifyBelief** *is called in any universe.* **EnterEvidence** *is associated with JTs.*

6 Added Operations for Regaining Global Consistency

As discussed in Section 4, parallel evidence entering at multiple agents renders the single-agent MSBN operation for maintaining global consistency invalid. In order to regain global consistency in a multi-agent LJF, we extend the inward-outward belief propagation method in a single junction tree [4] to a LJF. Jensen's method propagates belief through a single information channel (a unique path exists between any two cliques in a junction tree). Belief propagation in a LJF must be performed over multiple linkages. Fortunately, the latter problem has been solved in single-agent MSBNs with the operation **UpdateBelief**.

Following the above idea, we add two new operations **CollectNewBelief** and **CommunicateBelief** to regain global consistency in a multi-agent LJF. **CollectNewBelief** causes an inward belief propagation in the LJF. **CommunicateBelief** calls **CollectNewBelief** and **DistributeBelief** to propagate evidence obtained from multiple agents (JTs) inward first and then outward to the entire LJF.

Operation 4 (CollectNewBelief) *Let T be a JT in a LJF. Let* **caller** *be either the LJF or a neighbour JT. When* **CollectNewBelief** *is called in T, the following are performed: (1) T calls* **CollectNewBelief** *in all neighbours except* **caller** *if* **caller** *is a neighbour. (2) After each neighbour being called has finished* **CollectNewBelief**, *T updates its belief with respect to the neighbour by* **UpdateBelief**. **CollectNewBelief** *is associated with JTs.*

Operation 5 (CommunicateBelief) *When* **CommunicateBelief** *is initiated at a LJF F, the following are performed: (1) A JT T in F is arbitrarily selected. (2)* **CollectNewBelief** *is called in T. (3) When T has finished* **CollectNewBelief**, **DistributeBelief** *is called in T.* **CommunicateBelief** *is associated with the LJF.*

7 Consistency After and Between Communications

We answer the two questions raised in Section 4. Proofs of theorems are omitted due to the limited space. First, we show that the operations proposed in Section 6, when performed, guarantees global consistency among multiple agents:

Theorem 6 (Multi-agent consistency) *Let F be a supportive, globally consistent and separable LJF converted from a MSBN of hypertree structure. Let Z be a subset of JTs of F. After the following operations, F is globally consistent: (1) For each JT in Z, use* **EnterEvidence** *to enter finite pieces of evidence into the JT. (2) Use* **CommunicateBelief** *to communicate belief among JTs.*

CommunicateBelief involves global computation of multiple agents and information exchange over 'distance'. Due to the cost involved, **CommunicateBelief** can not be performed frequently. Therefore, each agent may acquire multiple pieces of evidence between two successive **CommunicateBelief** operations, and may have to answer queries before the next **CommunicateBelief** can be performed. The second question we address is: what is the consistency level of these answers to queries. Theorem 7 shows that, between two successive communications, a JT is consistent with all local evidence acquired so far, and is consistent with all global evidence acquired up to the last communication. This is the best that one can expect.

Theorem 7 (Semi-up-to-date) *Let F be a supportive and separable LJF converted from a MSBN of hypertree structure. Let Z be a subset of JTs of F.*

After a **CommunicateBelief** *in F followed by a finite number of* **EnterEvidence** *to each JT in Z, the marginal distributions obtained in a JT $T \in Z$ are identical as would be obtained if only the* **EnterEvidence** *operations in T were performed after the* **CommunicateBelief**.

8 Discussion

In a MSBN, the interface between any pair of subnets satisfies a d-sepset condition such that, to make neighbours up-to-date, it is sufficient to pass the up-to-date distribution on the interfacing set between them and nothing else.

Each subnet is transformed into a single JT. The tree structure maintains a single information path between any pair of nodes in the JT such that local belief updating can be performed efficiently. The price paid is the multiple paths between neighbour JTs: Passing evidence from one JT to a neighbour requires computation proportional to the number of linkages between them.

Such built-in preference of the efficiency of local computation over the efficiency of communication is consistent with the general assumption in multi-agent intelligent systems. In DAI, we assume that the distance between agents prevents constant communication. Therefore, each agent must rely on its local resource in its problem solving, and can communicate with other agent only occasionally.

One of the major concerns in DAI is how to balance the need to maintain as much as possible global consistency and the need to reduce the traffic of communication. As argued by Pearl [10], a tree structure makes use of causal independence and allows the most efficient information passage among a group of elements. Our study on MSBNs highlights the MSBNs that are organized into a *hypertree* structure. This structural preference is a more general case of the exploration of the causal independence in singly connected BNs [9] and in the JT representation of multiply connected BNs [4].

If we call an element which can render a pair of elements conditionally independent a 'causal mediator', we see an increase of complexity of the internal structures of causal mediators in the three cases. In singly connected BNs, a causal mediator is an internal node in the network. In the JT representation of multiply connected BNs, a causal mediator is an internal clique, a group of variables. In MSBNs/LJFs of hypertree structure, a causal mediator is a subnet/JT. As argued by Pearl [10], conditional independence should not be viewed as a restrictive assumption for mathematical convenience, nor as an occasional grace of nature for which we must passively wait, but rather as a mental construct that we should actively create. The progression of probabilistic reasoning techniques from singly connected BNs, to the JT representation of multiply connected BNs, and to MSBNs/LJFs is just one example of such endeavor.

References

1. A.H. Bond and L. Gasser. An analysis of problems and research in dai. In A.H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, 3–35, 1988.
2. A.H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.
3. L. Gasser and M.N. Huhns, editors. *Distributed Artificial Intelligence, Volume II*. Morgan Kaufmann, 1989.
4. F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Comput. Stat. Quarterly*, (4):269–282, 1990.
5. F.V. Jensen, K.G. Olesen, and S.K. Andersen. An algebra of bayesian belief universes for knowledge-based systems. *Networks*, 20:637–659, 1990.
6. S.L. Lauritzen and D.J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, (50):157–244, 1988.
7. V.R. Lesser and L.D. Erman. Distributed interpretation: a model and experiment. *IEEE Transactions on Computers*, C-29(12):1144–1163, 1980.
8. R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley, 1990.
9. J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, (29):241–288, 1986.
10. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
11. Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.
12. Y. Xiang, D. Poole, and M. P. Beddoes. Multiply sectioned bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9(2):171–220, 1993.

Building Bridges between Knowledge Representation and Algebraic Specification

Jacques Calmet
Universität Karlsruhe, IAKS
Postfach 6980
76128 Karlsruhe, Germany
calmet@dkauni2.bitnet

Indra A. Tjandra**
Concordia University, Dept. of CS
PQ, H3G 1M8
Montreal, Canada
ono@cs.concordia.ca

Abstract. An approach to transforming the algebraic specification of *a mathematical domain of computation* into a knowledge base, preserving the semantics determined in the specification, is introduced. It involves the algebraic specification language FORMAL-Σ and the hybrid knowledge representation system MANTRA. In the framework of FORMAL-Σ mathematical domains of computation are represented algebraically. The transformation aims at achieving the executability of a specification.

Keywords: Methodology, modeling mathematical domains of computation, knowledge representation, algebraic specification, program transformation

1 Introduction

Constructive type theory generally emerges to be an important discipline in programming languages. For computer scientists it provides a framework which brings together logic and programming languages in a most elegant and fertile way. Many "new" programming languages take this discipline into account. They provide the user with a capability to define an arbitrary abstract and concrete data type, type inheritance, and generic functions. Among these programming languages are: C++ [Stroustrup], HASKELL [Fasel et al.], and AXIOM [Sutor].

One deals essentially with type discipline in symbolic computing. One consideration, when designing or implementing an algebraic algorithm, is the validation of the algorithm according to a specific concrete data type, i.e. in which type the algorithm is valid. For example, an implementation of the Chinese remainder algorithm is valid solely for those arguments in the range of the data type Euclidean domain. This amounts to providing the user a framework for specifying the abstract data type Euclidean domain and for implementing a concrete one.

Abstract data types in symbolic computing are inherently modular. The operation symbols specified in a type possess certain properties. For these reasons, it turns out that one way to represent them consists in making use of an algebraic specification formalism. Consequently, in some existing symbolic computation systems specification languages are embedded. A well-known system having

[2] **This research was funded in part by Institute of Robotics and Intelligent Systems and by Natrural Sciences and Engineering Research Council of Canada

such a language is AXIOM. Basically, these systems suffer from being able to interpret the intended meaning of the specifications in the following sense. One can specify an abstract data type algebraically, e.g. by determining the sorts, the operation symbols, and their properties, but the properties are handled as comments, i.e. the intended meaning gets lost.

Knowledge representation systems, such as

- KL-TWO [Vilain],
- MANTRA [Bittencourt, Calmet-Tjandra-Bittencourt],
- KANDOR [PatelSchneider84], and
- KRYPTON [Brachman et al.]

are systems appropriate to be used to represent such properties. The major difficulty, coming up from using such a knowledge representation directly, consists in coding the specification of a data type into the formalism possessed by the system. This would presume the user to be very familiar with knowledge representation formalisms. Our work consists in building bridges between algebraic specification, which is on one hand very formal but not operational, and knowledge representation, which is on the other hand operational.

This paper is organized as follows. It is essentially necessary to develop an algebraic specification language which is appropriate to be used for these purposes. For this reason, we have developed the algebraic specification language FORMAL-Σ [Calmet-Tjandra]. We present its characteristical features in the next section. Our target language is embedded in the knowledge representation system MANTRA. In section 3 we introduce the formalisms used in Mantra and we describe the motivation of using MANTRA. Our approach, FORMAL-Θ, to transforming a specification into a knowledge base of MANTRA is introduced in section 4. This section aims at depicting a methodology to achieve executability of a FORMAL-Σ module. We omit technical details and proofs. Finally, some concluding remarks are presented in section 5. The reader is assumed to be familiar with basic notions of knowledge representation, algebraic specification, and program transformation.

2 FORMAL-Σ

Initially, we want to give an overview of the framework of *unified algebras* [Mosses]. Based on this framework we define the syntax and the semantics of FORMAL-Σ.

A unified signature Σ^u is a *homogeneous first-order signature*. In contrast to order-sorted signatures there is only one sort, $\mathcal{S}^u$, instead of a set of order-sorted sorts[3]. Let Σ^u be a pair $\langle \mathcal{S}^u, \Omega^u \rangle$, where $\Omega^u \supseteq \Omega^{u0} = \{\bot, |, \&\}$. $\bot$, $|$ and $\&$ represent the bottom of a lattice, the joint and meet operations on lattices, respectively. As we deal with homogeneous first-order signatures we can write $\Omega^u = \{\Omega^u_n \mid n \geq 0\}$, where Ω^u_n is the set of operator symbols of arity n.

[3] As we shall see below, we still can simulate an order-sorted signature using a unified domain, since the carrier of a unified domain is a distributive lattice.

Let X be a set of variables, disjoint from Ω^u. A Σ^u-unified property is a universal Horn Clause involving equalities with variables from X, operator symbols form Ω^u and the binary predicate symbols $=$, $\leq$, and $:$.

A Σ^u-unified algebra $\mathcal{D}$ is a homogeneous Σ^u-algebra such that:

(i) $|\mathcal{D}|$ is a distributive lattice with $\_|\_{\mathcal{D}}$ as join, $\_\&\_{\mathcal{D}}$ as meet and $\bot_{\mathcal{D}}$ as bottom. Let $\leq_{\mathcal{D}}$ be the partial order of the lattice.
(ii) There is a distinguished subset of incomparable values, $\mathcal{E}_{\mathcal{D}} \subseteq | \mathcal{D} |$.
(iii) For each $f \in \Omega^u$, the function $f_{\mathcal{D}}$ is monotone with respect to $\leq_{\mathcal{D}}$.

The intended interpretation of the binary predicate symbols, in a unified algebra $\mathcal{D}$ is as follows:

- $\mathrm{x} = \mathrm{y} :\Leftrightarrow$ x is identical to y
- $\mathrm{x} \leq \mathrm{y} :\Leftrightarrow \mathrm{x} \leq_{\mathcal{D}} \mathrm{y}$
- $\mathrm{x} : \mathrm{y} :\Leftrightarrow \mathrm{x} \in \mathcal{E}_{\mathcal{D}}$ and $\mathrm{x} \leq_{\mathcal{D}} \mathrm{y}$

The syntax and the semantics of FORMAL-Σ can be described as follows. A specification is represented by a module. The module concepts to build up large specification from smaller ones are illustrated by some examples (Figure 1). The Module `Boolean` (or `SemiGroup`) is specified as a *basic module* possessing the constants `Boolean`, `true` and `false` whose relationships are embodied by the relators "=", "|" and ":". It also possesses the function symbols `and`, `or` and `not` that are represented in the *Operation* part together with their functionalities. The properties of these function symbols are expressed by means of Horn Clauses with equality. The Module `Rng` and `Leftmodule` are specified by using the module concepts `Union` and `Rename`, and `Union`, `Rename` and a formal parameter `Rng`, respectively.

The symbol `-?>` is used to specify a partial function. The three relator symbols coincide with those as defined above.

`Use` and `Define` indicate the imported and the exported signatures of a module, respectively. The union of two modules is denoted by the module concept `Union`. It builds the union of the imported and exported signatures. It should be noted that ambiguities have to be avoided, e.g. name duplication. The module concept `Composition` offers a possibility to compose two modules. It is comparable to the concept `enrich` of CLEAR [Sanella] or `composition` of ACT-TWO [Ehrig-Mahr]. The composition of modules `m1` and `m2` takes the imported signature of `m2` as its imported signature and the exported signature of `m1` as its exported signature. The module concept `Rename` is used to rename only particular exported operation and constant symbols of a module. The module concept `Ignore` allows to get rid of some exported operation and constant symbols. Using this module concept one can simplify a module by forgetting or hiding some operation and constant symbols that, for instance, no longer need to be exported.

The semantics of a module `m` is given by the signature function $\mathcal{S}[\![m]\!]$ consisting of $\mathcal{S}_i[\![m]\!]$ in $Sign_{\mathcal{UNI}}$, the imported unified signature of `m`, and $\mathcal{S}_e[\![m]\!]$ in $Sign_{\mathcal{UNI}}$, the exported unified signature of `m`, and the meaning function $\mathcal{B}[\![m]\!]$ presenting a partial function $\mathcal{B}[\![m]\!] : Mod_{\mathcal{UNI}}(\mathcal{S}_i[\![m]\!]) \rightsquigarrow Mod_{\mathcal{UNI}}(\mathcal{S}_e[\![m]\!])$.

```
(Module
 Boolean
 (Define
  (Constants
   (= Boolean (| true false))
   (: true Boolean)
   (: false Boolean))
  (Operations
   (or ((Boolean Boolean) -> Boolean))
   (and ((Boolean Boolean) -> Boolean))
   (not (Boolean -> Boolean)))
  (Clauses
   (= (or true true) true)
   (= (or true false) false)
   (= (or false true) false)
   (= (or false false) false)
   (= (and true true) true)
   (= (and false true) false)
   (= (and true false) false)
   (= (and false false) false)
   (= (not true) false)
   (= (not false) true))))
```

```
(Module
 SemiGroup
 (Define
  (Constants
   SemiGroup)
  (Operations
   (o (SemiGroup SemiGroup) -> SemiGroup))
  (Clauses (Imply
              (: (a b c) SemiGroup)
              (= (o (o a b) c)
                  (o a (o b c)))))))
```

```
(Module
 Monoid
 (Union
  (SemiGroup)
  (Define
   (Constants
    (=< Monoid SemiGroup)
    (: Neutral Monoid))
   (Clauses (Imply
               (: a Monoid)
               (and (= (o Neutral a)
                          a)
                    (= (o a Neutral)
                         a)))))))
```

```
(Module
 Group
 (Union
  (Monoid)
  (Define
   (Constants (=< Group Monoid))
   (Operations (inv Group -> Group))
   (Clauses (Imply
               (: (a b) S)
               (and (= (op (inv a) a)
                          Neutral)
                    (= (op a (inv a))
                          Neutral)))))))
```

```
(Module
 AbelianGroup
 (Union
  (Group)
  (Define
   (Constants
    (=< AbelianGroup Group)
   (Clauses (Imply
               (: (a b) AbelianGroup)
               (= (o a b)
                   (o b a)))))))
```

```
(Module
 Rng      (*a help definition for Ring*)
 (Union
  (AbelianGroup
   (Rename (AbelianGroup Neutral o inv)
              (AddAbelianGroup 0 + -)))
  (SemiGroup
   (Rename (SemiGrouup o)
              (MultSemiGroup *)))
  (Define
   (Constants
    Rng)
   (Clauses (=< Rng AddAbelianGroup)
              (=< Rng MultSemiGroup)
              (Imply
               (: (a b c) Rng)
                  (and (= (* a (+ b c))
                            (+ (* a b) (* a c)))
                        (= (* (+ a b) c)
                            (+ (* a c) (* b c)))))))))
```

```
(Module
 LeftModule
 (Use Rng)
 (Union
  (AbelianGroup
   (Rename (AbelianGroup Neutral o inv)
              (AddAbelianGroup 0 + -)))
  (Define
   (Constants
              (=< LeftModule AddAbelianGroup))
   (Operations
              (* (Rng LeftModule) -> LeftModule))
   (Clauses (Imply
               (: (x y) Rng)
               (Imply
                (: (a b) LeftModule)
                (and (= (* x (* a b))
                          (* (* x a) b))
                     (= (* x (+ a b))
                          (+ (* x a) (* x b)))
                     (= (* (+ x y) a)
                          (+ (* x a) (* y a))))))))))
```

Figure 1: Simple Examples of Modules

3 MANTRA

We adopt the knowledge representation language of MANTRA to be used as the target language. In this section we describe the motivation of using MANTRA and its characteristical features.

MANTRA is a multi-layered system and its architecture is made up of three levels:

(i) **the epistemological level**, that is concerned with formalisms showing how facts about the world can be represented in the memory of a computer,
(ii) **the logical level**, that consists of a knowledge base management including primitives for storing and legitimating conclusions to be drawn from the facts stored in the knowledge bases, and
(iii) **the heuristic level**, which is concerned with mechanisms to search spaces of possible solutions, to resolve conflict situations, to match patterns and to give explanations if desired.

The decidability of all algorithms involved is achieved by adopting a four-valued semantics based on the works of Patel-Schneider [PatelSchneider90], Belnap [Belnap], and Thomason et al. [Thomason et al.].

The epistemological level consists of three modules: An assertional module, based on a decidable logic, a frame module, based on the terminological box of Krypton [Brachman et al.], and a semantic network module providing inheritance with exceptions [Etherington]. The primitives of these modules are used as parameters of the **Tell** and **Ask** primitives of the logical level. The **Tell** and **Ask** primitives are used to store facts and to interrogate knowledge bases, respectively.

The syntax of these two primitives, which can be regarded as commands, is the following:

command ::= tell(*knowledge base*, *Fact*) | ask(*knowledge base*, *Query*)

Fact ::= to-logic(*formula*) | to-frame(*frame-def*) | to-snet(*snet-def*)

Query ::= from-logic(*formula*) | from-frame(*frame-question*) |
from-snet(*snet-question*)| from-logic-frame(*logic-frame-question*) |
from-logic-snet(*logic-snet-question*) |
from-frame-snet(*frame-snet-question*)

where *knowledge base* is the name of a particular knowledge base.

The assertional module is intended to be used to represent assertional knowledge about a particular domain. The expressions of this language are first-order logic formulae. The reasoning of this logic is based on t-entailment as introduced in [PatelSchneider90].

The terminological module is intended to be used to represent a terminology by means of concepts, the categories of objects, and relations, the properties of objects. The notion of relations is an extension of the notion of roles, usually used in terminological languages. Roles are binary relations and relations are arbitrary n-place relations.

The semantic network module manipulates the notions of classes and hierarchies. The hierarchies can be explicitly created by defining links among classes. Two types of links are provided: *Default* links and *Exception* links. The hierarchies are used as inheritance paths between classes. The main inference procedure of this module calculates the *Subclasses* relation taking into account the explicit exception.

Finally, the third level consists of primitives allowing the definition of Horn Clauses where each clause may contain the primitives defined in the logical level. An SLD engine is used to interpret the clauses. The idea underlying the heuristic level is to allow the introduction of ad hoc rules in the inference process. These rules can directly introduce domain knowledge in the knowledge bases or they can specify strategies for the utilisation of the logical level.

4 FORMAL-Θ

Firstly, we sketch how to transform a specification. Accordingly, we shall give some transformation rules for transforming the carrier of a module. For the sake of simplicity we omit the proofs. The technical details, which are not suitable to be presented in this paper, can be found in [Tjandra].

A transformation is defined over the basic modules in the sense that modules possessing module constructs, e.g. `union`, must initially be converted to semantically equivalent basic modules by using the underlying semantic functions.

A specification is transformed into a knowledge base according to its syntax tree in a bottom up manner. Each transformation step makes use of a particular transformation rule represented in the following form:

$$\underset{0}{\overset{I}{\overline{\hspace{6em}\mathrm{R}\hspace{6em}}}}\quad\left\langle\begin{matrix}C_1\\ \cdots\\ C_n\end{matrix}\right.$$

The above form is also equivalent to the following one:

$$\frac{\{C_1, \cdots, C_n\}}{\vdash R[I, O]}$$

The intended meaning of such a rule is the following. I and O are called input and output scheme respectively. I is part of a specification in FORMAL-Σ and O is the corresponding representation of I in the language of MANTRA. A program scheme is a term from $W(PL \cup X)$, the term algebra over $PL \cup X$, i.e. a term over PL (programming language) containing free variables from a countable set X of typed scheme parameters. $C_1 \cdots C_n$ are applicability conditions which are Horn clauses over an enrichment of $PL \cup X$, i.e. they may contain additional syntactic and semantic predicates over program schemes.

A transformation rule is correct if it constitutes a valid inference, i.e. if the program schemes I and O are in the semantic relation indicated by R whenever the applicability conditions are valid.

The transformation of a specification can be outlined as follows:

- The signature, coinciding with the main construction of a specification consisting of the module identifier, formal parameters and the function symbols, is represented by means of frames provided at the epistemological level of MANTRA.
- The carrier, that is a distributive lattice (in a unified algebra), is also modeled by frames.
- The Horn clauses imposed on the specification is represented by Horn clauses in MANTRA at the heuristic level. As MANTRA does not allow equations we extend the approach to integrating functional programming into logic programming proposed by van Emden and Yukawa [VanEmden-Yukawa] in such a way, that equations can also be treated in MANTRA.

The correctness of the transformation of a specification, i.e. the specification and its corresponding representation in MANTRA are semantically equivalent, is verified by giving as proof for each transformation rule that there is a morphism from I to O according to the semantic predicate R.

The following transformation rules might give a rough idea on how to transform the carrier of a module, i.e. the lattice of a unified algebra, into its corresponding representation in MANTRA's formalism.

The rule $\mathcal{R}.\mathcal{M}eet$, $\mathcal{R}.\mathcal{L}t$, and $\mathcal{R}.\mathcal{J}oin$ are used to transform part of a module involving "&", "|", and "$\leq$", respectively, cf. section 2.

The rule $\mathcal{R}.\mathcal{L}t$ is used to transform part of a module in the form: (=< C_1 C_2), where C_1 and C_2 are program schemes and "$\leq$" is the relation as defined in section 2. The output of $\mathcal{R}.\mathcal{L}t$ involves the primitive **Tell**, cf. section 3.

The rule $\mathcal{R}.\mathcal{M}eet$ is applied to transform the following module part: (= C_1 (| C_2 $\cdots$ C_n))

$\mathcal{R}.\mathcal{M}eet$:

(= ⟨*C1*⟩(& ⟨*C2*⟩ ··· ⟨*Cn*⟩))

$R_{F^{\circ}H^{\circ}}$

```
(Tell 'C
      '(Define-Concept
        ⟨C1⟩
        (And-Concept ⟨C2⟩··· ⟨Cn⟩)
        (Forall Attribute K)))
```

M is a Module ∈ $\mathcal{M}odule$

C is a knowledge base

(= ⟨*C1*⟩ (& ⟨*C2*⟩ ··· ⟨*Cn*⟩)) is part of M

$$K := \begin{cases} M & ;\langle C1\rangle \in (DefConstants \quad M) \\ (Greatest\ \langle C1\rangle) & ;otherwise \end{cases}$$

The rule $\mathcal{R}.\mathcal{L}t$ is applied to transform the following module part: $(=<\ C_1\ C_2)$ with

$\mathcal{R}.\mathcal{L}t$:

(=< ⟨*C1*⟩ ⟨*C2*⟩)

$\overline{\qquad\qquad\qquad R_{F^o H^o} \qquad\qquad\qquad}$

```
(Tell '⟨C⟩
     '(Define-Concept
       '⟨C2⟩
       (Forall Attribute K2))

(Tell 'C
     '(Define-Concept
       ⟨C1⟩
       (And-Concept
        ⟨C2⟩
        (Forall Attribute K1))))
```

M a Module ∈ $\mathcal{M}odule$

(: ⟨*C1*⟩ ⟨*C2*⟩) is part of M

$\mathcal{C}$: Knowledge base

$$K1 := \begin{cases} M & ; C1 \in (DefConstants\ \ M) \\ (Greatest\ C1) & ; otherwise \end{cases}$$

$$K2 := \begin{cases} M & ; C2 \in (DefConstants\ \ M) \\ (Greatest\ C2) & ; otherwise \end{cases}$$

The rule $\mathcal{R}.\mathcal{J}oin$ is applied to transform the following module part: $(=\ \ C_1\ \ (\&\ \ C_2\ \ \cdots\ \ C_n))$ with

$\mathcal{R}.\mathcal{J}oin$:

(= ⟨*C1*⟩(| ⟨*C2*⟩ ··· ⟨*Cn*⟩))

$\overline{\qquad\qquad\qquad R_{F^o H^o} \qquad\qquad\qquad}$

```
(Tell 'C
     '(Define-Concept
       ⟨C1⟩
       (Or-Concept ⟨C2⟩··· ⟨Cn⟩)
       (Forall Attribute K)))
```

M is a Modul ∈ $\mathcal{M}odule$

C a knowledge base

(= ⟨*C1*⟩ (| ⟨*C2*⟩ ··· ⟨*Cn*⟩)) is part of M

$$K := \begin{cases} M & ; \langle C1\rangle\ \in (DefConstants\ \ M) \\ (Greatest\ \langle C1\rangle) & ; otherwise \end{cases}$$

5 Conclusion

We have given an overview of an approach to transforming a specification into a knowledge base. We make use of the algebraic specification language FORMAL-Σ as a source language and the language of the hybrid knowledge representation MANTRA as the target language. The main idea concerning the correctness of the transformation consists in determining a homomorphism between these two formalisms.

The systems, FORMAL-Σ, MANTRA, and FORMAL-Θ are components of a programming environment, called LEMMA, that is intended to be used in symbolic computing. The whole system, LEMMA, that is the integration of the existing components (as described above) and some other components that are not described in this paper (such as a learning component and a theorem prover), is currently being developed.

Although symbolic computing has been chosen as our application domain, the transformation approach proposed is generic enough to be applied in other application domains, e.g. to specify and to analyze software requirements as well as software reuse. In such an application domain a framework of algebraic specification is, usually, adopted for specification purposes and a knowledge representation formalism can be used to achieve executability of the specifications for analysis purposes.

References

[Belnap] N.D. Belnap. A useful four-valued logic. In J.M. Dunn and G. Epstein, editors, *Modern Use of Multiple-Valued Logics.* D. Reidel, 1977.

[Brachman et al.] R.J. Brachman, R.E. Fikes, and H.J. Levesque. KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, 16(10), pp. 67 –73, October 1983.

[Bittencourt] G. Bittencourt. *An Architecture for Hybrid Knowledge Representation.* PhD thesis, Universität Karlsruhe, Institut für Algorithme und Kognitive Systeme, 1990.

[Calmet-Tjandra] J. Calmet and I.A. Tjandra. A unified-algebra-based specification language for symbolic computing. In A. Miola, editor, *Design and Implementaion of Symbolic Computation Systems.* Springer-Verlag, LNCS 722, pp. 122 – 133, 1993.

[Calmet-Tjandra-Bittencourt] J. Calmet, I.A. Tjandra, and G. Bittencourt. MANTRA: A shell for hybrid knowledge representation. In E. Lee, B. Wah, N.G. Bourbakis, and W.T. Tsai, editors, *Tools for Artificial Intelligence*, pp. 164 – 171. IEEE Computer Society Press, 1991.

[Ehrig-Mahr] H. Ehrig and B. Mahr. *Fundamental of Algebraic Specification 2.* Monograph on Theoretical Computer Science, vol **21**. Springer-Verlag, 1990.

[Etherington] D.W. Etherington. *Reasoning with Incomplete Information: Investigation of Nonmonotonic Reasoning.* PhD thesis, University of British Columbia, Vencouver, CS, 1986.

[Fasel et al.] J.H. Fasel, P. Hudak, S.P. Jones, and P. Wadler. SIGPLAN notices special issue on the functional programming language HASKELL. *ACM Sigplan Notices*, 27(5):1, 1992.

[Mosses] P.D. Mosses. Unified algebras and institutions. In *Logics in Computer Science*, pp. 304 – 312. IEEE Press, 1989.

[PatelSchneider84] P.F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of Workshop on Principle of Knowldege-Based Systems*, pages 11 –19. IEEE, 1984.

[PatelSchneider90] P.F. Patel-Schneider. A decidable first-order logic for knowledge representation. *Journal of Automated Reasoning*, 6, pp. 361 – 388, 1990.

[Sanella] D Sanella. A set-theoretic semantics of CLEAR. *Acta Informatica*, 21(5), pp. 443 – 472, 1984.

[Stroustrup] B. Stroustrup. *The C++ Programming Language*. Addison Wesley, 2nd edition, 1993.

[Sutor] R.S. (Ed.) Sutor. AXIOM *User's Guide*. The Numerical Algorithm Group Limited, 1991.

[Thomason et al.] R.H. Thomason, J.F. Horty, and D.S. Touretzky. A calculus for inheritance in monotonic semantic nets. CMU-CS-86-138, Carnegie Mellon Univeristuy, Dept. of CS, 1986.

[Tjandra] I.A. Tjandra. *Algebraic Specification of Mathematical Domains of Computation and Type Polymorphisms in Symbolic Computing (in German)*. PhD thesis, University of Karlsruhe, Dept. of CS, 1993.

[Vilain] M. Vilain. The restriction language architecture of a hybrid representation system. In *Proceeding of 9th IJCAI*, pp. 547 – 551, 1985.

[VanEmden-Yukawa] M.H. VanEmden and K. Yukawa. Logic programming with equations. *The Journal of Logic Programming*, 4, pp. 265 – 288, 1987.

Turning an Action Formalism Into a Planner—Essentials of a Case Study

Joachim Hertzberg[1] and Sylvie Thiébaux[2]

[1] Dortmund University, Informatik VIII, 44221 Dortmund, Germany, and
GMD, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
[2] IRISA, Campus de Beaulieu, 35042 Rennes, France

Abstract. We describe notions that are useful for building planners whose reasoning about action matches a given formal calculus for such reasoning. As a planner has to reason under resource constraints, one may require that its plans are just approximately correct wrt. the action formalism. To nonetheless ground its practical reasoning formally, we develop the notion of limited correctness of a planner wrt. an action formalism.

> *Even if we were to grant that most of the advances in AI are inspired by experimentalists and that formalists serve mainly to "tidy up", it is nevertheless our opinion that the important new results in AI will be achieved by those researchers whose experiments are launched from the high platform of solid theory.* — Genesereth & Nilsson [6, p. viii]

1 Background

The purpose of an *action formalism* is to specify the reasoning about prerequisites and consequences of actions that an ideally rational agent should perform; among the many examples for such formalisms are the situation calculus [9], Pednault's [11] formalization for determining possibly context-dependent effects, and the possible-models-based formalism suitable for context-dependent and non-deterministic effects under incomplete information described in [2], which underlies our study. A *planner* has to do reasoning about prerequisites and consequences of actions, but if it insists on doing it in an ideally rational way, then it is likely to fail on most realistic usage constraints for practical applications. Consequently, a typical planner either does only trivial reasoning about action, or a neat formal description of its reasoning about action yields surprise [8], or it just has no neat description. In sum, it seems as if the camp of interesting planners and the camp of interesting action formalisms are way apart. However, this gap should be bridged, because not bridging it would imply that planning has to reinvent much of the work already done in action formalisms.

A natural idea for bridging the gap is to design planners whose plans are only *approximately* correct wrt. to a given action formalism. However, it is not completely clear what a useful approximation should be here: plausible alternatives are probabilistic approximations in analogy to PAC (<u>p</u>robably <u>a</u>pproximately <u>c</u>orrect) learning [14], and run-time dependent approximations that, in the limit, converge to correct reasoning. For this paper, we chose the second alternative, because it combines nicely with viewing planning as an anytime [3] activity, which, in turn, seems plausible under incomplete information and non-deterministic action effects as formalized in [2].

This paper summarizes a case study in which we have developed the planner PASCALE2, proceeding along the lines just sketched; the study is described comprehensively in a full paper [7]. For lack of space, we cannot present in any interesting detail particularities of the formalism that we used for the study, of the corresponding plan format, or of the resulting planner; the reader is referred to the full paper. We do present here the more general notions of building a planner starting from some action formalism. Note, however, that this is based on a generalization from one single complex case; so it may turn out that even if our notions and procedure were completely adequate, other formalisms should be approximated differently in planners.

Section 2 introduces the basic notions of action formalisms, planning problems, plans, and planners generating these plans. Moreover, the section introduces the notion of "strict" correctness and completeness of planners wrt. an action formalism. Section 3 defines and discusses limited planner correctness, the central concept for approximating the reasoning of an action formalism. Section 4 points to related work and concludes.

2 Action Formalisms, Problems, Plans, and Planners

To describe how to build a planner on the basis of an action formalism, we have to state more clearly what an action formalism and a planner are in the first place. As we do not want to go into the details of one particular action formalism here, we will only state their general aspects that are necessary for this paper. In particular, we do not really define what plans and planners are, but we only state requirements that we assume every concrete such object to fulfill; realistic plans and planners will include many more substructures or procedures than we address here, but the claim is that, e.g., any planner having at least those components that we require a planner to have, can be looked at in the way we describe.

We do *not* believe, let alone assume, that there is *the* one universal action formalism to rely on. The variance in the required expressivity is huge for different application domains, involving or not involving, resp., incompleteness of information, time, non-deterministic action effects, or other features. Different action formalisms intended for domains of different characteristics should mirror these differences, and so will the respective planners and their related concepts like planning problem and plan. Let us start with characterizing action formalisms.

At the level of abstraction required here, an action formalism is a formal apparatus that takes the description of a state of affairs in the world and the description of an action, and yields a description of (possible) *resulting* states of affairs. An example is the situation calculus a world state is described by a set of first order formulas, an action is represented by a syntactical function called operator, and the action effects are given by another (unique) world state where the resulting changes are axiomatized by first order formulas, too. Note that formalisms admitting for multiple alternative action effects may yield non-unique successor states: every formalism capable of expressing the action of tossing a coin is an example.

Practically, one might wish to require that the formalism be correct in the sense that, given an intuitively "right" description of the previous state and a "right" description of an action, it yields a "right" (set of) successor state description(s). However, this is no issue here: in the following, we assume that all action formalisms are correct in this sense. For simplicity, we restrict the language for describing world states to first order logic. Moreover, we assume a language for describing actions, every individual such description being called an *operator.* In the example of situation calculus, this action language is a first order language. An action formalism, then, consists of a language for describing situation, a language for describing actions, and a mapping that axiomatizes the result of action applications subject to the following constraints:

Requirement 1 (Result) *Given a first order language $\mathcal{L}$ and a language $\mathcal{A}$ for describing actions (each such description called an* operator*), the result of an operator is described by a function $r : \mathcal{L} \times \mathcal{A} \to 2^{\mathcal{L}}$.*

Turning now to the planning side of the story, the "classical" definition of a planning problem, and, accordingly, of a plan for solving it, requires that a problem consist of start state, goal description, and a set of operators. We slightly generalize this view:

Requirement 2 (Planning problem description) *Let $\mathcal{L}$ be a first order language and $\mathcal{A}$ an operator language. A planning problem description over $\mathcal{L}$ and $\mathcal{A}$ (or problem, for short) is a quadruple $\langle s, g, K, O\rangle$, where $s \in \mathcal{L}$ describes the initial situation (possibly incompletely), $g \in \mathcal{L}$ describes the goal, $K \subseteq \mathcal{L}$ is the background knowledge, and $O \in \mathcal{A}$, the operator inventory, is a set of operators.*

It is useful to identify background knowledge as that part of the domain description which does not change by any application of actions; examples are causal laws or descriptions of terminological equivalences.

It sometimes helps to include an initial plan in a problem description that may be used as the starting point of a planner, where some planning techniques might start with a plan dummy consisting of pseudo-operators representing descriptions of the initial and final states, and other others might start with a plan skeleton or a finished library plan that have to be properly instantiated or modified. Trying to keep separated the notions of problem and plan, we do not follow this pragmatically useful idea. Remember that this and the following requirements are just meant to specify constraints for the respective notions that get used later in this text; concrete theories or implementations may add other components as they need.

Classically, a plan is a set of occurrences of operator instances with a (possibly partial) order imposed on them, and we will keep these requirements. Note, however, that we do not require that the order be interpreted in the usual, "classical" STRIPS [5] way or that actions have unique effects: in the plan format in [7], we allow,e.g., for actions with multiple alternative effects. The additional requirement has to do with how to determine operator results in a plan.

Classical planning assumes the result of an operator to be independent from the situation in which it is applied, facilitating to determine these results looking at the operator alone. In general, this is too simple, and we need a more elaborated means to tell what the effects or set of possible effects are of an operator in the plan. How this is determined has to do with the underlying time model (are operator executions allowed to overlap? Is the effect of overlapping action execution different from the union of operator effects?), with the plan representation (are effects explicitly specified in the plan?), and possibly other factors. On the other hand, there *must* be some such means, because it must be possible to determine whether some operator in the plan has its preconditions satisfied and whether the goal description is fulfilled. Consequently, we require that there be some function that, given a plan and one of its operators, tells what this operator effects in; this is described by a set of formulas, corresponding to possible effect alternatives yielded by the operator itself or by different contexts it can be applied in. This leads to the following requirements for plans:

Requirement 3 (Plan) *Let $\Psi = \langle s, g, K, O \rangle$ be a problem over $\mathcal{L}$ and $\mathcal{A}$. A plan for Ψ is a triple $\langle \mathcal{I}, \Omega, \prec \rangle$, where*

- *Ω is the set of operator occurrences, i.e., elements of Ω are elements of O, but identical operators may occur more than once and are distinguishable;*
- *$\mathcal{I}$ is the initial node, i.e., a pseudo- (non-executable) operator meant to (possibly incompletely) describe the initial state. For notational convenience, we require that $\mathcal{I} \in \Omega$.*
- *$\prec$ is the operator order, i.e., an ordering relation on Ω, such that $\mathcal{I}$ is the unique $\prec$-smallest element.*

Let plan *and* operator *denote the types of plans and operators in plans, resp.*

There is a function r_p : plan × operator $\longrightarrow 2^{\mathcal{L}}$ *returning the result of an operator in a plan.*[3]

Note that the operator ordering may be interpreted differently, depending on the time model that is used.

These requirements fulfilled, we can now define the notions of (strict) correctness and completeness of a *plan* wrt. an action formalism and a problem description Ψ. The intuition behind plan *correctness* is to say that if a plan Π says that something is or may be the case, then it really is or may be the case. In particular, if the plan execution proceeds as a plan tells it, then Π, using r_p, will rightly predict what may result, according to the action formalism's r, from applying an action; every action it directs to execute is in fact executable, and if it directs to stop the plan execution, then a situation will have emerged in

[3] For notational convenience, we identify the type plan with the *set* of plans here; analogously for operators.

which Ψ's goal formula is true. (All this can, of course, only be guaranteed for a plan, if the planning problem description itself was intuitively correct wrt. the real world domain).

The intuition behind plan *completeness* is that if something may happen at plan execution according to Ψ and the action formalism, then the plan does already respect it. In particular, the plan start must subsume the initial situation, and the plan must contain all results of actions that it directs to apply. Both plan correctness and completeness can be defined rigorously, but we do not do so here for lack of space. In the full paper [7], we have given the respective definitions for the particular type of plans used there.

A correct and complete plan is the ideal plan that one would like a planner to generate, namely, a "solution" to the planning problem, and these two requirements are the background for the vast majority of papers on planning—albeit mostly implicit. We make them explicit here in order to deal with controlled relaxations of them; this will be the key for achieving limited correctness of planners in the next section.

The notions of correctness and completeness apply analogously to *planners,* not only to plans. In our context, a planner is an entity that, given a planning problem description, returns a plan after a number of computation steps. These steps may be time ticks, plan expansion steps or more generally, any monotonically increasing time function. For simplicity, we assume a discrete measure over the natural numbers here. Moreover, we assume the existence of types natural and problem description containing objects that are natural numbers and problem descriptions, respectively. Under the theoretical view that we are having here, a planner is then constrained as follows:

Requirement 4 (Planner) *A planner is a pair* $\langle \mathcal{P}, A_{\mathcal{P}} \rangle$ *of functions, where* $\mathcal{P}$: problem description $\rightarrow$ natural $\rightarrow$ plan *is the* plan generation function *and* $A_{\mathcal{P}}$: problem description $\rightarrow$ natural set, *the* availability function, *is a total function such that* $\mathcal{P}(\Psi)$ *is a total function on* $A_{\mathcal{P}}(\Psi) \subseteq \mathbb{N}$ *for every problem* Ψ.

Intuitively, $\mathcal{P}$ is the planning procedure itself, and $A_{\mathcal{P}}$ defines the set of steps at which $\mathcal{P}$ has a plan for a problem available. An obviously interesting special case is a planner for which $A_{\mathcal{P}} = \mathbb{N}$, i.e., a planner that has an output available after any number of steps—an *any-step planner,* as one could call it. In fact, this is the direct analog to the now-famous anytime planners as first discussed in [3].

The intuitive idea behind planner correctness or completeness, then, is simple: A planner is *correct* wrt. some criterion to be specified, if all the plans it delivers for a planning problem meet this criterion; and it is *complete,* if it eventually generates all plans for the problem that meet some other to-be-specified criterion. The obvious question is what these ominous criteria are supposed to be. In general, they can specify everything you like. The criterion that we will hard-wire into the definitions of both planner correctness and completeness is supposed to be useful, at least as long as we remain on the theory side: it is—somewhat unsurprisingly—plan correctness and completeness. That means that a correct planner will deliver only correct and complete plans, and that a complete planner will eventually deliver all correct and complete plans for a given problem.

Definition 1 (Planner correctness). A planner $(\mathcal{P}, A_{\mathcal{P}})$ is correct iff the plan $\mathcal{P}(\Psi)(t)$ is correct and complete wrt. Ψ for all problems Ψ and for all $t \in A_{\mathcal{P}}(\Psi)$.

Definition 2 (Planner completeness). A planner $(\mathcal{P}, A_{\mathcal{P}})$ is complete iff, for all problems Ψ and for all plans Π that are correct and complete wrt. Ψ, there exists $t \in A_{\mathcal{P}}(\Psi)$ s.t. $\mathcal{P}(\Psi)(t) = \Pi$.

Note that it is very easy to design planners that are *either* correct *or* complete. An example for a trivially correct planner is one that returns no plan at all; all plans it returns are correct and complete. An example for a trivially complete planner is one that enumerates all plans: It will eventually also generate all correct and complete plans. Hence, the interesting matter is to design planners that are correct *and* complete.

At least, that is what pure theory tells. As exposed in the introduction of this paper, the issue here is to develop planners that are correct *in the limit;* for this task, the "ideal" correctness as just defined will just serve as our reference point. Defining this more liberal version of correctness is what we will do now.

3 Correctness, Ltd.

In view of our goal to achieve planners that are only correct in the limit relative to an action formalism, we allow, but do not require that such a planner use a "direct" implementation of the formalism. Instead, one may use a restriction, or a formalism implementation that in itself only approximates the formalism's result function r. It is only the planner whose behavior we constrain: In the limit, it must be correct; and meanwhile, its incorrectness must be describable relative to the ideal, as objectified by the formalism. However, we cannot exemplify this any further here, given that this paper is formalism-independent; [7, Sec. 4.1] gives an example.

The main tool for expressing limited correctness is a rating function for incomplete and incorrect plans that, given a plan, determines the "degree" of its correctness and completeness, supplementing the sharp notions of correctness and completeness with a gradual valuation for plans that are not correct and complete—i.e., the vast majority of plans that planners practically deal with. Normalizing the values of rating functions to the real interval $[0, 1]$, we get:

Definition 3 (Rating function). Let Ψ be a planning problem description. A rating function for Ψ is a total function ϱ_Ψ mapping plans for Ψ to $[0, 1]$, such that for every plan Π: $\varrho_\Psi(\Pi) = 1$ iff Π is correct and complete wrt. Ψ

In the rest of the paper, we sloppily speak of some ϱ as a *family* of rating functions for a problem *domain* with potentially many problem descriptions, to express that ϱ_Ψ is a rating function for every problem Ψ in the domain.

The notion of correctness in the limit is based on the following idea. Standard notions of planner correctness and completeness (Definitions 1 and 2) require that a planner generate all correct and complete plans for a problem, and only these. Practically, that is too restrictive. In particular, as long as nothing is assumed about preferences among correct and complete plans, it is even worthless to generate more than one such plan: one is as good as all the others. Furthermore, if the number of computation steps available to generate plans is small, then even incomplete and incorrect plans may be of interest. Using a rating function for measuring the "degree" of plan correctness and completeness, we consequently require that (1) the plans delivered get more correct and more complete as run time increases, and that (2) a correct and complete plan is returned eventually at some step t. Of course, we do not assume that a limitedly correct planner is given as much as t steps, nor that it stops running then: it may continue, searching for plans that are not only correct and complete, but also maximize some other plan quality function.

Definition 4 (Planner correctness in the limit). Let ϱ be a rating function family. A planner $\langle \mathcal{P}, A_{\mathcal{P}} \rangle$ is limitedly correct wrt. ϱ, iff for all problems Ψ:

1. for all $t_1, t_2 \in A_{\mathcal{P}}(\Psi)$, if $t_2 \geq t_1$ then $\varrho_\Psi(\mathcal{P}(\Psi)(t_2)) \geq \varrho_\Psi(\mathcal{P}(\Psi)(t_1))$, and
2. if there is a correct and complete plan wrt. Ψ, then there exists $t \in A_{\mathcal{P}}(\Psi)$ s.t. $\varrho_\Psi(\mathcal{P}(\Psi)(t)) = 1$

Comparing this definition with the one of "strict" planner correctness (Definition 1), a remarkable theoretical difference is that limited correctness requires that at least one correct and complete plan be returned, whereas strict correctness does not. This may be interpreted as a flavor of "completeness in the limit" in this concept of limited correctness. The definition guarantees, on the other hand, that a limitedly correct planner actually turns to a correct planner after the magic number of t steps, for which there is no analogy with respect to completeness. Even with this idea in mind, the monotonicity condition (item 1) is more restrictive than might be considered necessary: another reasonable variant of limited correctness could require planner correctness after the first correct and complete plan has been delivered, but not insist on monotonicity.

However, the very idea of planner correctness in the limit makes sense only if plan correctness and completeness is the dominant factor in an overall measure of plan quality; and given this, it seems reasonable to require a non-arbitrary, predictable behavior of the planner output with respect to this factor. There may be additional criteria to discriminate different correct and complete plans, measured by other quality sub-functions. We neither require nor exclude these.

Note that all correct planners that eventually return plans for all solvable problems, i.e., $A_{\mathcal{P}}(\Psi) \neq \emptyset$ for all solvable Ψ, are limitedly correct wrt. to all rating functions—as intuitively expected. Consequently, all correct and complete planners are limitedly correct wrt. to all rating functions. Note further that incorrect, incomplete, and both incorrect and incomplete planners may be limitedly correct wrt. some rating function. This is of most importance here. For good reasons, practical planners tend to be highly incorrect and incomplete. The notion of correctness in the limit allows them to keep this salvaging property, yet being put into a clear relationship to an arbitrarily neatly or idealistically defined action formalism. Of course, being able to establish this relationship does not come for free. It still requires designing an appropriate rating function and building the planner accordingly. However, we are convinced that this effort is worth being taken, if the goal is to build planners delivering plans with a clear semantics, yet operating under realistic run time constraints.

4 Related Work and Conclusion

So what have we achieved? There are two ideas behind this work, which are basically independent, but fit together well: First, the output of a planner should be describable relative to some rigorously defined action formalism; second, approximating the formalism is sufficient for this describability, given realistic constraints on a planner's resources. Both these ideas are not new, but combining them seems uncommon. Let us briefly put them into the perspective of the literature and then conclude.

There is an increasing amount of work providing logical formalizations of planning systems or of their basics, e.g., [8, 11]. The rationale is that formally rigorous descriptions can help understand the systems, their fundamental procedures, limitations and theoretical complexity—and that formal description can simply serve as a more efficient tool than natural language for communicating a system's essentials, as other sciences have experienced before.

There are two ways to provide a planner with a neat formalization. The first is the *ex post* way: The planner comes first, and the attempt to formalize it, later; Lifschitz's [8] semantics of STRIPS [5] is the most prominent example, and it also demonstrates that the planner implementors may have used tricks that require a matching formalization to be more intricate than expected. The second is the *ex ante* way: The basic formalization comes first, and the planner is implemented later; an example for that direction is Pednault's [11] formalization for determining possibly context-dependent (i.e., conditional) action effects using regression, with a planner implementation provided by McDermott [10].

The *ex post* strategy—while perfectly adequate when the field of planning was in its infancy—does not appear wise for planning as a maturing engineering sub-field: It gives you no guidance as to from which theory platform to start with your action representation and calculus.[4] If STRIPS operators were an adequate action representation for all interesting application domains, then there would be no problem, [8] telling you all formal background you need; but most planning researchers seem to agree that they are not. Consequently, we need a blueprint for using a given operator formalism in a to-be-designed planner.

A somewhat traditional method for going this *ex ante* way is to do *deductive planning* [1]. If you can afford to use a planner that is correct and complete in the sense of Definitions 1 and 2 (or in some other "strict" sense), and if you have a suitable deductive theorem prover at hand, then this is probably the method of choice. However, assuming that these conditions are not universally true, the

[4] Obviously, it gives you no guidance either as to how to write your planner, e.g., which search strategy and heuristics to employ. But this is a different point. Here, we have to do with the *representation* aspects.

blueprint must look different, and we have proposed to base it on the idea of a planner *approximating* the action formalism in the sense of Definition 4. Within AI research, this has its roots back in Simon's [13] idea of ***bounded rationality;*** more recently, Russell's group has worked on *limited rationality* (e.g., [12]). Another related idea is anytime planners [3], focusing on the requirement that planning has to be a timely activity. Our contribution is to connect the ideas of reasoning within limited resources and of grounding a planner's reasoning about action in a pre-specified action formalism.

Space not permitting to describe more than just the most general framework for this connection, let us finally sketch at least the most characteristic features of the planner PASCALE2 that we have developed according to the notions just introduced [7]. Starting point is the action formalism described in [2] that allows

- information about the world to be incomplete in the sense that it may be only known that $p \vee q$ is the case (for arbitrary formulas p, q) or that it is not known whether p or $\neg p$ is true;
- actions to have alternative effects, the minimalistic standard example being the action of tossing a coin whose effect is either Heads or Tails; and
- actions to have context dependent effects, the standard example being to toggle a light switch, resulting in lights On if Off before, and vice versa.

Moreover, [7] allows probability information to be added to the domain representation, describing, e.g., that, lacking other information, the light is Off with a 40% chance, or that tossing an (unfair) coin yields Heads with a 70% chance.

Plans matching this formalism must be highly conditional, and the effects of operators not unique, so classical nonlinear plans are inappropriate. Assuming that the language $\mathcal{L}$ for describing the domain is finite, the respective plan format is much like a finite non-deterministic automaton: The nodes correspond to finite descriptions of world states; every such state has an operator associated; and the arcs correspond to the possible outcomes of applying this operator. Additionally, the probability information allows arcs to be labeled with transition probabilities. Executing such a plan means to react with the specified action, once the plan executor finds itself in a world state matching a specified description.

In sum, PASCALE2 has many features of modern decision-theoretic anytime planners much like,e.g., the one described in [4], yet is based on the safe platform of a well-understood action formalism. The probabilistic information makes it particularly intuitive to define a rating function using decision-theory: the degree of *plan* correctness and completeness is the expected cumulative value of the *operators* in that plan (this value expressing the overlap between the r and r_p functions). Just to give a flavor of the behavior of a limitedly correct planner, Figure 1 shows the values of PASCALE2's rating function (called CCD) over run time on a demo problem explained in [7].

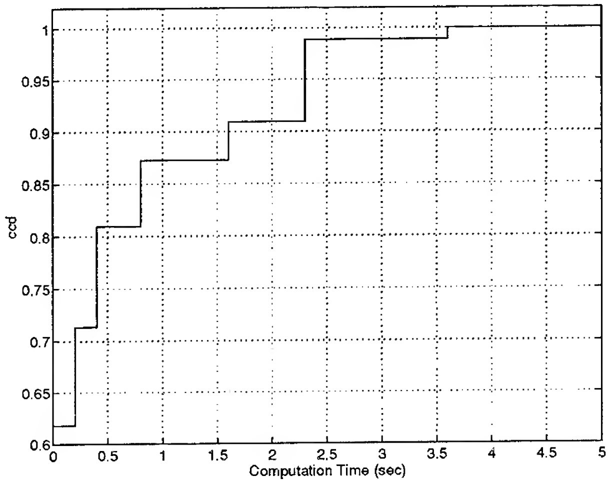


Fig. 1. Plan quality as a function of computation time on a demo problem [7].

References

1. S. Biundo. Present-day deductive planning. In C. Bäckström and E. Sandewall, editors, *Current Trends in AI Planning. EWSP'93 – 2nd European Workshop on Planning*, pages 1–5. IOS Press, 1994.
2. G. Brewka and J. Hertzberg. How to do things with worlds: On formalizing actions and plans. *J. Logic and Computation*, 3(5), 1993.
3. T.L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. AAAI-88*, pages 49–54, 1988.
4. T.L. Dean, L. Kaelbling, J. Kirman, and A. Nicholson. Planning with deadlines in stochastic domains. In *Proc. AAAI-93*, pages 574–579, 1993.
5. R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to theorem proving in problem solving. *J. Art. Intell.*, 2:189–208, 1971.
6. M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Palo Alto, CA, 1988.
7. J. Hertzberg and S. Thiébaux. Turning an action formalism into a planner—a case study. *J. Logic and Computation (Sp. Is. Actions and Processes)*, 4(5), Oct. 1994.
8. V. Lifschitz. On the semantics of STRIPS. In M.P. Georgeff and A.L. Lansky, editors, *Proc. 1986 Workshop Reasoning about Actions and Plans*, pages 1–9, Los Altos, 1987. Morgan Kaufmann.
9. J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–507, 1969.
10. D. McDermott. Regression planning. *Int. J. Intell. Syst.*, 6(4):357–416, 1991.
11. E.P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *J. Computational Intelligence*, 4:356–372, 1988.
12. S. Russell and E. Wefald. *Do the Right Thing. Studies in Limited Rationality*. MIT Press, Cambridge, Massachusetts, 1991.
13. H. A. Simon. *Administrative Behavior. A Study of Decision-Making Processes in Administrative Organizations*. Free Press, New York, 3 edition, 1976. Original: New York (Macmillan) 1947.
14. L. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.

Towards Refinement of Definite Logic Programs[*]

Jan Komorowski and Silvia Trcek[**]

Knowledge Systems Group
Department of Computer Systems and Telematics
The Norwegian Institute of Technology, The University of Trondheim
N-7034 TRONDHEIM, NORWAY

Abstract. A refinement calculus for deriving logic programs is proposed. The calculus has a model-theoretic and a transformational characterization. The transformational characterization is based on partial deduction and is shown to be sufficient to preserve the model-theoretic characterization. Refinement with partial deduction can be thus considered as a tool for the step-by-step derivation of logic programs. Refinement rules formalize the so called "Laws of Programming" and thus they constitute a basis for a mechanical system that can approximate human knowledge of programming.

1 Introduction

We propose a calculus for refining logic programs such that a theory (defined by a logic program) can be designed incrementally, that the design steps reflect rules of programming, that each design step is correct, and that the design is monotone. Monotonicity is considered an essential design property because it allows replacement of parts of a theory by refinements of the parts. We give theorems that establish a model-theoretic characterization of refinement and provide a syntactic connection to refinement by partial deduction. The theoretical work is complemented by an implementation of a workstation-based, graphical environment that supports partial deduction of logic programs [15].

Related work Not surprisingly, formal methods for development of logic programs have been evolving in a different direction than methods for imperative programming. Transformational approaches are well developed in logic programming (see, for example, Clark and Tärnlund, [3], Komorowski, [7, 8], and Gallagher [6] and more recently Sato and Tamaki [20]), while in imperative programming a refinement calculus was developed by Back [1].

[*] This research is supported in part by the ESPRIT BRA COMPUNET/NFR contract # 469.92/011 and by the Human Capital and Mobility NFR contract # 101341/410.

[**] On leave of absence from Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe (TH).

Although the idea of a refinement calculus seems quite useful for logic programs, there is no comparable work done for that paradigm, as far as we know. Some related contributions are: [17, 2, 19, 12, 18]. The work presented here has been motivated by our own suggestion that partial deduction could be a suitable framework for refinement of logic programs [9]. However, we gave no formal framework there.

Structure of the paper A general definition of refinement is presented first. We then give an overview of the s-semantics. Section 4 introduces the main theorems that characterize refinement of logic programs. Section 5 provides refinement operators. The final discussion is concerned with refinement rules and experimental derivations. This article is highly condensed due to space restrictions. Consequently, no proof of the theorems or lemmata are provided here. The reader can consult [16], which was a preliminary version of this paper. It is also assumed that the reader is acquainted with the basic notions of logic programming and partial deduction. For an overview of partial deduction see, for example, [13]. For a recent selection of research in partial deduction see [14] and [21]. However, we give a short overview of the s-semantics [5] on which our refinement calculus is based.

2 Refinement of Programs

Intuitively, refinement is a successive transformation of a program (specifications are treated as programs in this context) into another program while preserving some desired properties of the initial program. Refinement calculus provides us with a tool that ensures that each step from the initial specification to the program will be done in a correct way. Moreover, given a refinement in one context we would like it to be true also in other contexts, since it is likely that a refined program will be embedded in another one.

The concept of correct refinement has been defined, for example, by Back in [1]. It can be defined for any class of programs and specifications as follows:

Definition 1 Refinement. Let S and S' be programs. S is correctly refined by S', denoted by S **ref** S', if S' satisfies any specification that S does, i.e. S sat $R \Rightarrow S'$ sat R for any R in the set of specifications.

From the definition it immediately follows that the refinement relation **ref** is a pre-order (i.e. it is reflexive and transitive). The satisfaction relation **sat** is usually a kind of correctness criterion. Another aspect is the importance of monotonicity: only if the constructs for combining programs into larger ones are monotonic with respect to (hence abbreviation wrt) the refinement relation, will refinement be useful in program derivation. Monotonicity in this case means that a sub-program T in a program $S[T]$ can always be replaced by its refining program T':

Definition 2 Monotonicity. Let $S[T]$ be a program containing sub-program T. S is monotone wrt **ref** if T **ref** $T' \Rightarrow S[T]$ **ref** $S[T']$.

Monotonicity depends, however, on the class of programs which should be combined as well as on the class of specifications and the definition of the satisfaction relation **sat**.

3 Semantics

The standard Herbrand semantics is not suitable for refinement because it is difficult to compare programs such as $p(X)$ and its refinement $p(a)$. The s-semantics [5] that we review below seems to be an appropriate choice, also for this purpose. This semantics is based on interpretations that contain also non-ground atoms. To be able to use orderings on such interpretations we need an equivalence relation that is defined below.

A *relation* $\leq$ *on atoms* can be defined by $A \leq A'$ iff there exists a substitution θ such that $A\theta = A'$. We say that A is less instantiated (more general, less specific) than A'. $\leq$ is a pre-order; we denote the induced equivalence relation (renaming) by $\approx$. The renaming relation can be generalized in an obvious way to apply to clauses. The renamed clause is sometimes called a *variant.*

Auxiliary notions A function on atoms that returns the predicate name of the atom is denoted *pred*(.). We will also need the following definitions.

Definition 3 A-closedness. Let S be a set of first order formulae or a set of atoms and $\mathbf{A}$ a finite set of atoms. S is called $\mathbf{A}$-*closed* if the following holds for all atoms $B \in atoms(S)$: if $\exists A \in \mathbf{A}\ pred(B) = pred(A)$ then $\exists A' \in \mathbf{A}\ A' \leq B$.

Hence, S is $\mathbf{A}$-*closed* if each atom in S containing a predicate symbol occurring in an atom in $\mathbf{A}$ is an instance of an atom in $\mathbf{A}$.

A set of atoms $\mathbf{A}$ is independent, if no pairs of atoms in $\mathbf{A}$ are instances of each other.

Definition 4 Independence of a set of atoms. The set of atoms $\mathbf{A}$ is called independent iff $\forall A \in \mathbf{A}$: if $\exists A' \in \mathbf{A}\ A \leq A'$ or $A' \leq A$ then $A = A'$.

3.1 Interpretations

The (new) *Herbrand base* $\mathbf{B}$ is defined as the quotient set of all the atoms (constructed from the predicates, functors (including constants) and variables) wrt $\approx$. The equivalence class of an atom A is, for simplicity, denoted by A itself. The atoms in $\mathbf{B}$ are ordered by $\leq$ in an obvious way. An *interpretation* is any subset of $\mathbf{B}$.

The following abstraction operations can be defined on interpretations. Upward closure: $Up(I) = \{A \in \mathbf{B} \mid \exists A' \in I\ A' \leq A\}$; Ground atoms: $Ground(I) = \{A \in I \mid A \text{ is ground }\}$; Minimal elements: $Min(I) = \{A \in \mathbf{B} \mid \forall A' \in I \text{ if } A' \leq A \text{ then } A = A'\}$. Intuitively, the upward closure of an interpretation I consist of all atoms in the Herbrand base that are more instantiated than those in I. The ground set of an interpretation is the corresponding standard Herbrand interpretation. The minimal elements are the most general representatives of an atom in I.

Ordering Relations on Interpretations

Definition 5. Let I_1, I_2 be interpretations. The following ordering relations can be defined: $I_1 \ll I_2$ iff $\forall A_1 \in I_1 \; \exists A_2 \in I_2$ such that $A_2 \leq A_1$ and $I_1 \underline{\ll} I_2$ iff $(I_1 \ll I_2)$ and $(I_2 \ll I_1$ implies $I_1 \subseteq I_2)$.

For an illustration of the definition consider the following example.

Example 1 Ordering relations. If $I_1 = \{p(a,b), p(a,X)\}$ and $I_2 = \{p(X,Y)\}$ then $I_1 \ll I_2$. If $I_1 = \{p(X,Y)\}$ and $I_2 = \{p(X,Y), p(a,a)\}$ then $I_1 \underline{\ll} I_2$.

Lemma 6. *Let $\mathcal{I}$ be a set of interpretations.*

- *$\ll$, defined on $\mathcal{I}$, is a pre-order (reflexive and transitive).*
- *$\underline{\ll}$, defined on $\mathcal{I}$, is an order (pre-order + antisymmetric).*

Lemma 7. *For any $I \in \mathcal{I}$, $I \subseteq Up(I)$.*

Lemma 8. *For any $I_1, I_2 \in \mathcal{I}$: $I_1 \ll I_2$ iff $Up(I_1) \subseteq Up(I_2)$.*

The ordering relations induce two equivalence relations on the set of interpretations $\mathcal{I}$.

Lemma 9. *Let $\equiv|_{\ll}$ and $\equiv|_{\underline{\ll}}$ be the induced equivalence relations of $\ll$ and $\underline{\ll}$, respectively. Then $I_1 \equiv|_{\ll} I_2$ iff $Up(I_1) = Up(I_2)$, and $I_1 \equiv|_{\underline{\ll}} I_2$ iff $I_1 = I_2$*

Lemma 10. *Let $I_1, I_1', I_2, I_2' \in \mathcal{I}$ be interpretations such that $\langle I_1, I_2' \rangle$ and $\langle I_1', I_2 \rangle$ are independent. If $I_1 \underline{\ll} I_1'$ and $I_2 \underline{\ll} I_2'$ then $I_1 \cup I_2 \underline{\ll} I_1' \cup I_2'$.*

3.2 Truth and Models

We can now define the notions of truth and models following [5]. The definitions are appropriate extensions of the standard notions which are based on ground interpretations.

Definition 11 s-truth. Let $I \in \mathcal{I}$ be an interpretation.

- A (possibly non-ground) atom A is s-true in I iff $\exists A' \in I \; A' \leq A$.
- A definite clause $A \leftarrow B_1, \ldots, B_n$ is s-true in I iff for each $B_1', \ldots, B_n' \in I$, if $\theta = mgu((B_1, \ldots, B_n), (B_1', \ldots, B_n'))$, then $A\theta \in I$.

In particular, a fact A is true in I iff $A \in I$, and an atom A is true in I if $A \in Up(I)$.

The definition extends the standard definition of truth in a (ground) Herbrand interpretation. When I is a standard Herbrand interpretation then $I = Up(I)$ and the first case in the definition reduces to A is true in I iff $A \in I$.

Definition 12 s-model. A model of a logic program P is any interpretation M in which all the clauses of P are s-true.

Lemma 13. *The intersection of all* s*-models of a program P is an* s*-model. In particular, it is the minimal* s*-model.* $\mathcal{M}(P) = \bigcap_i\{I_i \in \mathcal{I} \mid I_i$ *is an* s*-model of* $P\}$

Definition 14 *s*-semantics. The *s*-semantics of a definite logic program P is the minimal *s*-model $\mathcal{M}(\mathcal{P})$ of P.

Definition 15 Operational Semantics. Let P be a program. The computed answer substitution semantics of P is $\mathcal{O}(P) = \{A \mid \exists p \in \mathit{Pred}, \exists X_1, \ldots, X_n \in \mathit{Var}, \exists \theta \leftarrow p(X_1, \ldots, X_n) \xmapsto{P,\theta} \Box A = p(X_1, \ldots, X_n)\theta\}$

Theorem 16. *The operational semantics and the* s*-semantics of a program P are equivalent:* $\mathcal{O}(\mathcal{P}) = \mathcal{M}(\mathcal{P})$.

Example 2. Let P: c_1: `p(X, Y) ← q(X), r(Y).` c_2: `q(X).` c_3: `q(a).` c_4: `r(b).` The interpretation $I = \{r(b), q(a), q(Z), p(a, b), p(W, b), p(X, Y)\}$ is an *s*-model of P, but it is not the minimal one since there exists an interpretation I' with $I' \subseteq I$, e.g. $I' = \{r(b), q(a), q(Z), p(a, b), p(W, b)\}$. This model is minimal, $I' = \mathcal{M}(P) = \mathcal{O}(P)$. To see this, note that because of c_2, c_3, and c_4, the first three atoms have to be in $\mathcal{M}(P)$. Further, because $q(a) \in \mathcal{M}(P)$ and $r(b) \in \mathcal{M}(P)$, $p(a, b)$ has to be in $\mathcal{M}(P)$. Similarly, $p(W, b) \in \mathcal{M}(P)$.

In the coming sections, we use the notation $\mathcal{O}(\mathcal{P})$ to denote the semantics of the program P. Whenever we consider it convenient, we switch between the declarative and the operational representation of the semantics.

4 Refinement of Definite Logic Programs

We now formalize the notion of refinement for definite logic programs. Two definitions are given, one using the semantics in the sense of Sect. 3.2, the other with partial deduction. The first definition should be seen as the fundamental one. Based on the general definition in Sect. 2, it provides a model-theoretic characterization of refinement which is derived from the property that every logic program has an *s*-model.

The notion of A-refinement is introduced for refinement with partial deduction. A-refinement is dependent on a set of atoms **A**. The atoms in **A** are parameters which are used to control the program derivation. It is shown that A-refinement is a sufficient condition to preserve the model-theoretic refinement relation. Since A-refinement is a transformational concept it enables us to perform refinement steps syntactically. If the set of atoms **A** is restricted to so called "general atoms", A-refinement even preserves refinement equivalence in the sense of the semantical definition.

4.1 The Basic Notions

We define the satisfaction relation of Sect. 2 as P **sat** S iff S is a minimal *s*-model of P. This property is captured by the ordering relation $\lll$ given in Sect. 3. Furthermore, the equivalence induced by $\lll$ on programs says that two programs have the same minimal *s*-model.

Definition 17 Refinement of Definite Logic Programs. Let P, P' be definite logic programs. Then P **ref** P', denoted by $P \preceq P'$, iff $\mathcal{O}(P') \underline{\ll} \mathcal{O}(P)$.

Since $\underline{\ll}$ is an ordering, the induced equivalence of the refinement relation is a real equality (modulo variable renaming).

Definition 18 Refinement equivalence. Let P, P' be logic programs. P and P' are refinement equivalent, denoted by $P \equiv P'$, iff $P \preceq P'$ and $P' \preceq P$.

Theorem 19. *$P \equiv P'$ iff $\mathcal{O}(P) = \mathcal{O}(P')$*

Example 3. The programs P_1 and P_2 are refinement equivalent. P_1: c_1: `p(X, Y)` $\leftarrow$ `q(X), r(Y).` c_2: `q(X)` $\leftarrow$ `r(X).` c_3: `q(a).` c_4: `r(a).` P_2: c_1: `p(a, a).` c_3: `q(a).` c_4: `r(a).` Obviously, $\mathcal{O}(P_1) = \{r(b), q(a), p(a, a)\} = \mathcal{O}(P_2)$.

4.2 Refinement with Partial Deduction

The model-theoretic approach to refinement is rather impractical. With partial deduction refinement of programs can be carried out by syntactical transformations.

Definition 20 A-refinement. Let P, P' be definite logic programs. P is A-refined by P', denoted by $P \preceq_{\mathbf{A}} P'$, if there is a partial deduction P' of P wrt a set of atoms $\mathbf{A}$ such that P' is $\mathbf{A}$-closed and $\mathbf{A}$ is independent.

If the set $\mathbf{A}$ contains only one atom, $\mathbf{A} = \{A\}$, then we write $P \preceq_A P'$ instead of $P \preceq_{\{A\}} P'$.

Example 4. Let Q be given as follows: Q: c_1: `p(X)` $\leftarrow$ `q(X).` c_2: `q(a)` $\leftarrow$ `r(b).` $Q \preceq_{\mathbf{A}} Q'$ with $\mathbf{A} = \{p(a)\}$ and Q': c'_{11}: `p(a)` $\leftarrow$ `q(a).` c_2 : `q(a)` $\leftarrow$ `r(b).`

The refinement definition of A-refinement ensures that if the initial program is correct wrt its intended standard Herbrand model I then I is also a model of the new program. It is possible, however, that not every element in I is specified by the new program. Hence, the new program can be insufficient wrt I.

The following Theorem shows that A-refinement is a pre-order.

Theorem 21. *Let P, P_0, P_1, P_2 be logic programs.*

(1) There is a set of atoms $\mathbf{A}$ such that $P \preceq_{\mathbf{A}} P$.
(2) If there is a set of atoms $\mathbf{A}$ such that $P_0 \preceq_{\mathbf{A}} P_1$ and a set of atoms $\mathbf{B}$ such that $P_1 \preceq_{\mathbf{B}} P_2$ then there exists a set of atoms $\mathbf{C}$ which can be derived from $\mathbf{A}$ and $\mathbf{B}$ such that $P_0 \preceq_{\mathbf{C}} P_2$.

4.3 The Relation of Refinement and A-refinement

We now show that refinement with PD is a sufficient condition to preserve the model-theoretic refinement relation. Refinement equivalence can be achieved when the atoms wrt which the refinement is performed are "general" wrt the program. The following theorem states that A-refinement implies refinement. As a result, whenever we want to prove that two programs are in the refinement relation, we can show that they are in the A-refinement relation. A partial deduction wrt a set of atoms is a proper refinement only when the atoms are independent.

Theorem 22. *Let P, P' be logic programs, A an atom, $pred(A) = q$, and $\mathbf{A}$ an independent set of atoms. Then $P \preceq P'$ if $P \preceq_A P'$*and *$P \preceq P'$ if $P \preceq_{\mathbf{A}} P'$.*

Refinement equivalence is preserved if the atom A that determines the refinement process is general concerning P. The condition that P is $\{A\}$-closed ensures this kind of generality. Intuitively, no information is added by an atom that is more general than all the corresponding atoms (with the same predicate symbol) in the program.

Theorem 23. *Let P, P' be logic programs, A an atom, and $\mathbf{A}$ an independent set of atoms. If P is $\{A\}$-closed (correspondingly, $\mathbf{A}$-closed) then $P \equiv P'$ if $P \preceq_A P'$*(correspondingly, *$P \equiv P'$ if $P \preceq_{\mathbf{A}} P'$*).

Corollary 24. *Let P, P' be logic programs and $\mathbf{A}$ an independent set of atoms. If $\mathbf{A} \subseteq GenAtoms$ then $P \equiv P'$ if $P \preceq_{\mathbf{A}} P'$.*

Monotonicity The constructors of combining logic programs consist only of set-union. We cannot expect that monotonicity can be obtained in general since the $\mathfrak{s}$-semantics, like the Herbrand semantics, is not compositional.

Monotonicity wrt $\preceq$ The operator union is monotone regarding the refinement relation $\preceq$, if the programs do not have any predicate symbols in common.

Theorem 25. *Let P and Q be logic programs such that $preds(P) \cap preds(Q) = \emptyset$. Then $P \preceq P'$ and $Q \preceq Q'$ implies $P \cup Q \preceq P' \cup Q'$.*

Monotonicity wrt $\preceq_{\mathbf{A}}$ The condition for union to be monotone wrt $\preceq_{\mathbf{A}}$ is the same as the one in the previous paragraph.

Theorem 26. *Let P and Q be logic programs such that $preds(P) \cap preds(Q) = \emptyset$. Then $P \preceq_{\mathbf{A}} P'$ and $Q \preceq_{\mathbf{B}} Q'$ implies $P \cup Q \preceq_{\mathbf{A} \cup \mathbf{B}} P' \cup Q'$.*

Replacement of a sub-program by its A-refinement is monotone if the atoms in $\mathbf{A}$ only do contain predicate symbols occurring in the sub-program and if the intersection of the original program and the sub-program is A-closed. The restriction on the wrt-atoms is not a real one, since $T \preceq_{\mathbf{A}} T'$ is equivalent to $T \preceq_{\mathbf{A} \cup \mathbf{B}} T'$ where $\mathbf{B}$ is a set of atoms whose predicate symbols do not occur in T. It is required that the replaced program part is a sub-program, which means that, starting from any definition in the sub-program, all "reachable" procedures have to be contained in the sub-program.

Theorem 27. *Let $P[T]$ be a program containing sub-program T and T' some other program. Let $\mathbf{A}$ be a set of atoms. If $P - T$ is $\mathbf{A}$-closed then $T \preceq_{\mathbf{A}} T'$ implies $P[T] \preceq_{\mathbf{A}} P[T']$.*

The condition that the replaced program part is a sub-program is essential. It is easy to give an example showing that the theorem does not hold if arbitrary subsets of the program are A-refined. Furthermore, A-refinement is not monotone when the resulting program is not closed regarding the set of atoms that occur in the partial deduction.

Relation of A-refinement and the Herbrand Model Let us investigate how a special model, the least Herbrand model, behaves in the refinement process. For definite logic programs the following theorems can be obtained.

Theorem 28. *Let P, P' be definite logic programs. If $P \preceq_{\mathbf{A}} P'$ then $P \models P'$.*

Theorem 29. *Let P, P' be definite logic programs and M_P, $M_{P'}$ be their least Herbrand models respectively. If $P \preceq_{\mathbf{A}} P'$ then $M_P \supseteq M_{P'}$.*

Theorem 29 expresses that the least Herbrand model of a refining program can be reduced compared to the the least Herbrand model of the initial program. This enables us to set A-refinement in a closer context to the correctness of the program. Correctness of a program P is often defined as equivalence of the intended model I_P and the least Herbrand model M_P (see e.g. [4]). If $M_P - I_P \neq \emptyset$, i.e. the Herbrand model specifies more elements than the intended model, the program is said to be incorrect. If $I_P - M_P \neq \emptyset$, i.e. the least Herbrand model does not specify some elements which are in the intended model, the program is called insufficient. Hence, starting from an incorrect program it is possible to derive a correct program. A sequence of programs can be developed where the initial one describes some "big world", e.g. the Herbrand base, and the least Herbrand model of the final program equals the intended model. If the initial program is insufficient, however, the refining programs are also insufficient.

5 Refinement Rules for Logic Programming

From Theorem 27 we see that sub-programs can be replaced by an A-refinement. This means that we have to replace the (entire) definition of a predicate, if it is demanded that A-refinement be preserved. Other levels of "granularity" of replacements are sometimes useful. The idea is to provide operators that work on the level of the replacement of single atoms or clauses. Replacement should be understood in a general way such that, for instance, a replacement of an atom or a clause by 'nothing' is possible and has the meaning of deletion. These replacements cannot be justified by monotonicity of A-refinement or refinement since single clauses are (normally) not sub-programs, and atoms are not even subsets of a program.

We have investigated a number of operators and found, not surprisingly, that our opening and abbreviating tactics [10] and [11], as well as the tactics of Bossi and Cocco [2], preserve refinement equivalence. However, it is clear that addition or deletion of clauses or atoms can only be performed in a particular context.

5.1 Definitions of the Operators

unfold and *fold* are in some sense an inverse of each other. Roughly speaking, unfold allows to replace an atom in the body of a clause by a conjunction of atoms, whereas fold abbreviates a conjunction of atoms. *prune* and *add* make it possible to delete or to add a clause in a program. These operators work on the level of programs. On the clause level, *fatten* and *thin* can be named, allowing to add or delete an atom in the body of a clause.

Definition 30 unfold. Let P be a program, $c : A \leftarrow \widehat{A_{1,i-1}}, A_i, \widehat{A_{i+1,n}}$ a clause in P. Let c_j, $1 \leq j \leq m$ be all the clauses in P where there exists $\theta_j = mgu(head(c_j), A_i)$, $c_j : B^j \leftarrow \widehat{B^j_{1,h}}$. Define $c'_j : (A \leftarrow \widehat{A_{1,i-1}}, \widehat{B^j_{1,h}}, \widehat{A_{i+1,n}})\theta_j$. Then $unfold(P, c, A_i) = (P - \{c\}) \cup \{c'_j | 1 \leq j \leq m\}$.

Definition 31 fold. Let P be a program, $c : A \leftarrow \widehat{B_{1,i}}, \widehat{A_{1,k}}, \widehat{B_{i+1,n}}$ and $d : B \leftarrow \widehat{A'_{1,k}}$, $k \geq 1$, be clauses in P. Let $\theta = mgu(\widehat{A_{1,k}}, \widehat{A'_{1,k}})$. Define $c' : A \leftarrow \widehat{B_{1,i}}, B\theta, \widehat{B_{i+1,n}}$. Then $fold(P, c, \widehat{A_{1,k}}) = (P - \{c\}) \cup \{c'\}$.

Definition 32 prune (and add). Let P be a program, c a clause in P, then $add(P, c) = P \cup \{c\}$ (and $prune(P, c) = P - \{c\}$).

Definition 33 thin. Let $c : A \leftarrow \widehat{A_{1,i-1}}, A_i, \widehat{A_{i+1,n}}$ be a clause, then $thin(c, A_i) = A \leftarrow \widehat{A_{1,i-1}}, \widehat{A_{i+1,n}}$.

Definition 34 fatten. Let $c : A \leftarrow \widehat{A_{1,n}}$ be a clause and B an atom, then $fatten(c, B) = A \leftarrow \widehat{A_{1,n}}, B$.

5.2 Applicability Conditions and a Sample Development

PD is a highly indeterministic program transformation method. Usually many different residual programs exist for a PD of a program P wrt a goal G. PD is controlled by the computation rule and the choice of the wrt-atom. It is not only a problem when to stop the derivation, the order of the atoms selected is also essential to find a desired resultant. Hence, even if an operator can be characterized by a refinement with PD, it is usually not obvious how to prove that the operator is applicable. For some of the operators, a heuristic can be given which reduces the search space. The lack of space unfortunately prohibits a discussion of the applicability rules (for details see [16]).

Example 5 Post-order Evaluation. Suppose that a program for evaluating arithmetic expressions represented as binary trees is to be developed. Assume that an arithmetic expression is a number, a sum of two arithmetic expressions or a product of two arithmetic expressions (recursive definition).

Numbers are represented using successor arithmetics, 0 stands for zero and $s^{n+1}(0) = s^n(s(0))$, $n \geq 0$, stands for the natural number $n+1$. The binary tree is a node with a number or a tree with two branches and an operator. The Herbrand universe of the program is defined as follows: $U_{number} = \{0\} \cup \{s^n(0) \mid n \geq 1\}$, $U_{operator} = \{+, *\}$, $U_{tree} = \{tree(Left, op(Op), Right) \mid Left, Right \in U_{tree}, Op \in U_{operator}\} \cup \{node(X) \mid X \in U_{number}\}$, and, finally, $U_P = U_{number} \cup U_{operator} \cup U_{tree}$. To solve the problem a procedure *evaluate*/2 is developed such that its first argument is to be instantiated with a binary tree and its second argument is the value of the arithmetic expression. Using a divide-and-conquer algorithm schema the starting program is:

```
P1 : dac(Data, Sol) ← base(Data, Sol).
     dac(Data, Sol) ← divide(Data, D1, D2, D3),
          dac(D1, S1), dac(D2, S2), dac(D3, S3), compose(S1, S2, S3, Sol).
     evaluate(Expression, Val) ← dac(Expression, Val).
     base(node(X), X).base(op(+), +).base(op(*), *).
     divide(tree(Left, Operation, Right), Left, Operation, Right).
     compose(LVal, +, RVal, Val) ← plus(LVal, RVal, Val).
     compose(LVal, *, RVal, Val) ← times(LVal, RVal, Val).
```

Applying the operators of the refinement calculus results in the program:

```
P : evaluate(node(X), X).
    evaluate(tree(Left, op(+), Right), Val) ←
         evaluate(Left, LVal), evaluate(Right, RVal), plus(LVal, RVal, Val).
    evaluate(tree(Left, op(*), Right), Val) ←
         evaluate(Left, LVal), evaluate(Right, RVal), times(LVal, RVal, Val).
```

6 Conclusions

We have defined principles for a refinement calculus in logic programming. The calculus is an attractive approach to program derivation. A model-theoretic characterization can be realized syntactically by partial deduction. The calculus is monotone and thus it supports incremental development of programs because parts of a program (theory) can be replaced by their refinements. The refinement operators reflect rules of programming. A programming environment that supports sequences of refinements has been implemented [15]. Interesting issues to follow are: a development of a programming language based on the the refinement calculus, a relationship between the refinement calculus, inductive synthesis and abductive updates and other, both syntactic and semantic, refinement criteria.

References

1. R. J. R. Back. A calculus of refinements for program derivations. *Acta Informatica*, 25:593–624, 1988.
2. A. Bossi and N. Cocco. Basic transformation operations for logic programs which preserve computed answer substitutions of logic programs. *Journal of Logic Programming*, 16:47–87, 1993.
3. K. Clark and S.-Å. Tärnlund. A first-order theory of data and programs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 419–420, 1977.

4. W. Drabent, S. Nadjm-Tehrani, and J. Małuszyński. The use of assertions in algorithmic debugging. In ICOT, editor, *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 573–581, 1988.
5. M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. A model-theoretic reconstruction of the operational semantics of logic programs. *Information and Computation*, 102(1):86–113, 1993.
6. J. Gallagher. Transforming logic programs by specialising interpreters. In *ECAI-86. 7th European Conference on Artificial Intelligence, Brighton Centre, United Kingdom*, pages 109–122, 1986.
7. J. Komorowski. *A Specification of An Abstract Prolog Machine and Its Application to Partial Evaluation.* PhD thesis, Department of Computer and Information Science, Linköping University, Linköping, 1981.
8. J. Komorowski. Partial evaluation as a means for inferencing data structures in an applicative language: a theory and implementation in the case of Prolog. In *Proc. of the ACM Symp. Principles of Programming Languages*, pages 255–267. ACM, 1982.
9. J. Komorowski. Elements of a programming methodology founded on partial deduction - part 1. In Z. Ras, editor, *Proc. of the Int. Symp. on Methodologies for Intelligent Systems.* North Holland, October 1990. Knoxville, Tennessee.
10. J. Komorowski. Towards a programming methodology founded on partial deduction. In *Proc. of the European Conference on Artificial Intelligence.* Pitman Publ. Co., August 1990.
11. J. Komorowski. An introduction to partial deduction. In *Proc. of the META'92 Workshop on Meta Programming.* Lecture Notes in Computer Science, 1992.
12. J. Komorowski. On data-type centered, correctness-oriented design. In J. Jacquet, editor, *Constructing Logic Programs.* John Wiley & Sons, 1993.
13. J. Komorowski. A prolegomenon to partial deduction. *Fundamenta Informaticae*, 18(1):41–64, January 1993.
14. J. Komorowski. Special issue on partial deduction. *Journal of Logic Programming*, 16, 1993. Guest Editor.
15. J. Komorowski. PAL – a programming environment for the refinement calculus. Technical report, The Norwegian Institute of Technology, 1994.
16. J. Komorowski and S. Trcek. On refinement of logic programs. In *Proc. of the 5th Nordic Workshop on Program Correctness, Turku, Finland.* Åbo Akademi University, 1993.
17. A. Lakhotia. Incorporating 'programming techniques' into Prolog Programs. In E. Lusk and R. A. Overbeek, editors, *Proc. of the North American Conference on Logic Programming*, pages 426–440. MIT Press, 1989.
18. K.-K. Lau and S. Prestwich. Synthesis of a family of recursive sorting procedures. In *Proc. of the International Conference on Logic Programming*, pages 641–658. MIT Press, 1991.
19. A. Pettorossi and M. Proietti. Decidability results and characterization of strategies for the development of logic programs. In *Proc. of the International Conf. on Logic Programming*, Lisabon, Portugal, 1989. MIT Press.
20. T. Sato and H. Tamaki. First-order compiler: A deterministic logic program synthesis algorithm. *J. Symbolic Computation*, 8:605–627, 1989.
21. D. S. Warren. Memoing for logic programs. In *Special Issue of the CACM on Logic Programming.* ACM, March 1992.

AMPHION: Automatic Programming for Scientific Subroutine Libraries

Michael Lowry, Andrew Philpot, Thomas Pressburger, and Ian Underwood

AI Research Branch, NASA Ames
Recom Technologies, M.S. 269-2
Moffett Field, CA 94035

Abstract. This paper describes AMPHION[1], a knowledge-based software engineering (KBSE) system that guides a user in developing a formal specification of a problem and then implements this specification as a program consisting of calls to subroutines from a library. AMPHION is domain independent and is specialized to an application domain through a declarative domain theory. A user is guided in creating a diagram that represents the formal specification through menus based upon the domain theory and the current state of the specification. The diagram also serves to document the specification. Program synthesis is based upon constructive theorem proving, and is efficient and totally automatic.

1 Introduction

Subroutine libraries are one of the most prevalent forms of software reuse, particularly within the scientific programming community. However, users seldom have the time or inclination to fully familiarize themselves with even well-documented libraries. The result is that most users lack the expertise to properly identify and assemble the routines appropriate to their applications. This represents an inherent knowledge barrier that lowers the utility of even the best-engineered software libraries: the effort to acquire the knowledge to effectively use a library is often perceived as being more than the effort to develop the code from scratch. In domains with mature subroutine libraries, intelligent systems technology can greatly improve the productivity and quality of software engineering by automating the effective use of those libraries.

This paper describes a methodology for constructing an intelligent system that guides a user in creating a formal problem specification, and then automatically generates a program for this specification composed of subroutines from a library. The objective is to enable users who are familiar with the basic concepts of an application domain to program at the level of abstract domain-oriented problem specifications, rather than at the detailed level of subroutine calls. The domain-independent part of this methodology has been implemented in the AMPHION system, which includes generic specification acquisition and program synthesis subsystems. AMPHION is applied to an application domain by developing a declarative domain theory through the methodology described in this paper.

This methodology will be described by presenting AMPHION's application to the domain of solar system kinematics. (AMPHION applications in the domains of numerical aerodynamic simulation and space shuttle flight planning are currently under development.) A domain theory was developed that includes an abstract formalization of the domain suitable for expressing problems, and also includes the knowledge needed to

[1] Amphion, son of Zeus, played his magic lyre to charm the stones around Thebes into position to form the city's walls.

implement solutions to these problems using JPL's SPICELIB subroutine library. SPICELIB provides a tool kit for planetary scientists to construct programs analyzing the geometry of science observations for interplanetary missions. Typical problems include eclipses, occultations, moon shadows, and illumination angles. SPICELIB subroutines provide access to planetary ephemerides (the positions and velocities of planets as a function of time), spacecraft trajectories, and operations in analytic geometry. Complicating factors include the necessity of light-time correction over astronomical distances, and a plethora of formats and representations for locations, directions, and time.

AMPHION has undergone substantial testing with planetary scientists and has been installed at JPL for alpha testing in preparation for distribution to the planetary science community. New users are able to use AMPHION after a one hour tutorial. It usually takes a user an order of magnitude less time to develop a specification with AMPHION than to write and debug the corresponding program. In particular, experienced AMPHION users develop specifications in five minutes that take the SPICELIB developers an hour to code manually. New AMPHION users can develop a specification within fifteen minutes that take new users of SPICELIB a couple of days to develop a correctly working program. Further productivity gains are achieved through specification reuse and modification: the abstract graphical notation makes it much easier to identify the required modifications in a diagrammatic specification than it is to trace through dependencies in code. AMPHION's editing operations facilitate making the required modifications. Furthermore, there is no possibility of introducing bugs in the code, since AMPHION generates a new program from scratch for the modified specification. The specification in Figure 2 was generated in a few minutes by modifying a previous specification. The minor modifications to the specification resulted in substantially different programs, illustrating the advantages of specification modification versus program modification. These productivity improvements are consonant with the anticipated benefits of a specification-based software-engineering life cycle envisioned in [1].

AMPHION uses deductive synthesis [7] to routinely generate programs in this domain consisting of dozens of subroutine calls in under three minutes of CPU time on a Sun Sparc 2. Over a hundred programs have been generated to date. Because of the deductive synthesis, the programs are guaranteed to be correct implementations of users' specifications with respect to the domain theory.

To date, AMPHION has demonstrated the following capabilities essential for real-world KBSE: Users without training in formal methods readily develop domain-oriented diagrams denoting formal specifications using AMPHION's specification acquisition tools. Users can reuse, modify, and maintain previously developed specifications; thereby elevating software evolution from the code level to the specification level. Automatic deductive program synthesis achieves acceptable performance, given an appropriately structured domain theory and moderate use of theorem-proving tactics.

Section 2 of this paper presents an overview of the AMPHION system. Section 3 presents an example problem, its specification, and the program synthesized by AMPHION. Section 4 describes the domain-engineering methodology for the domain theory and overviews key steps in the example program synthesis. Section 5 describes AMPHION's generic specification acquisition subsystem. Section 6 summarizes the main points of the paper and compares it to previous work. The theorem-proving component is described in [9]. The theorem-proving tactics that make deductive program synthesis tractable for the specialized task of subroutine composition as well as an empirical analysis of program synthesis performance are presented in [6].

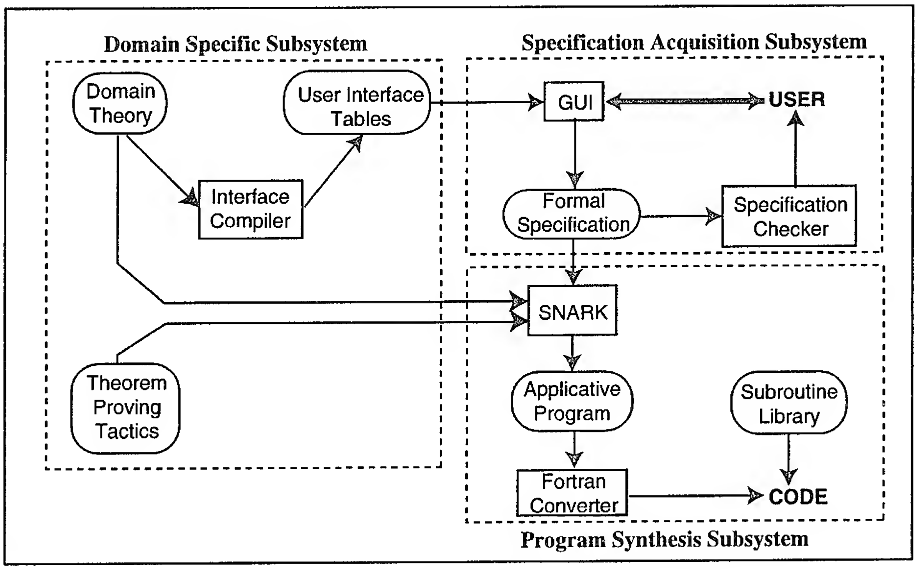


Figure 1: Flow diagram of AMPHION.

2 AMPHION System Overview

Figure 1 presents a flow diagram of AMPHION, where the dotted lines enclose subsystems, the rectangles enclose major components, and the rounded boxes enclose information. AMPHION is applied to a new domain by defining a domain theory and theorem-proving tactics. The domain theory is automatically translated into tables that drive the graphical user interface. The domain theory together with the theorem-proving tactics are used by the SNARK first order logic (FOL) theorem prover [9] both to check a specification and also to generate an applicative program. These three sources of information — the domain theory, derived user interface tables, and theorem-proving tactics — constitute the domain specific subsystem of an AMPHION application.

The graphical user interface (GUI) and the specification checker constitute the specification acquisition subsystem. A diagram developed interactively with the GUI is an alternate surface syntax for a formal problem specification in FOL augmented with the lambda calculus. *Lambda* is used for binding input variables, while the constructive existential quantifier *find* is used for binding output variables. Diagrams are equivalent to specifications of the following form (more general specifications must currently be entered textually):

lambda (inputs) find (outputs) exists (intermediates)
conjunct1 & .. & conjunctN

where each conjunct is either a constraint, *P(v1,..,vm)* , or an equation defining a variable through a function application, *vk = f(v1, .., vm)*.

AMPHION checks a specification by attempting to solve an abstracted version of the problem. If AMPHION cannot solve the abstracted problem, it employs heuristics to localize errors in the specification and give the user appropriate feedback. For example, if an output or intermediate variable cannot be solved in terms of the input variables, then that variable is under-constrained.

The program synthesis subsystem consists of an applicative program generator and a translator into the target programming language (e.g., FORTRAN-77 for the JPL SPICELIB subroutine library). After a valid specification is developed, it is converted into a theorem to be proved. The input variables of the specification are universally quantified and the output variables are existentially quantified within the scope of the input variables. An applicative program is synthesized through constructive theorem proving. During a proof, substitutions are generated for the existential variables through unification and equality replacement. The substitutions for the output variables are constrained to be terms in an applicative language whose function symbols correspond to the subroutines in a library.

The terms for the output variables are then translated into the target programming language through program transformations written in REFINE™ [8]. One set of transformations turns common subexpressions into lambda-bound variables in nested lambda applications. Another set of transformations handles subroutines with multiple outputs. Only the very last stage of the translation is programming-language specific: variable declarations and the sequence of subroutine calls are generated in the syntax of the target language. Targeting other programming languages would only require minor modifications.

3 Example Problem

The following problem will be used in this paper to illustrate the structure of the domain theory and key steps in program synthesis: A planetary scientist working on the Galileo mission to Jupiter wants to compute the solar incidence angle at the point on Jupiter's surface at the center of the boresight of an instrument on Galileo. The solar incidence angle is the angle between the surface normal and the apparent position of the sun. This problem is formalized as follows within the domain theory (variable names in italics):

Let *Solar-Incidence-Angle* be the angle between two rays, *SurfaceNormal* and *Ray-Intersection-Sun*.

Let *SurfaceNormal* be the ray normal to *Jupiter-Body* at the point *Boresight-Intersection*.

Let *Ray-Intersection-Sun* be the ray from the point *Boresight-Intersection* to *Sun-Body*.

Let *Boresight-Intersection* be the intersection point of the ray *Boresight* and *Jupiter-Body* .

Let *Boresight* be the ray from the location of Galileo-Orbiter at time *TGalileo* in the direction of *Direction-2*.

Let *Direction-2* be the direction pointed by the instrument *Galileo-Camera* at time *TGalileo*.

Let *Jupiter-Body* be Jupiter at time *TJupiter*.

Let *Sun-Body* be the Sun at time *TSun*.

Let *Photon-Sun-Jupiter* be a photon from *Sun-Body* to *Jupiter-Body*.

Let *Photon-Jupiter-Galileo* be a photon from *Jupiter-Body* to Galileo-Orbiter arriving at time *TGalileo*.

Let the representation of *Solar-Incidence-Angle*, the output, be in radians.

Let the representation of *TGalileo*, one input, be a string in the format for Galileo's internal clock.

Let the representation of *Galileo-Camera*, the other input, be an integer which identifies a particular instrument on Galileo-Orbiter.

Each of these sentences corresponds to a conjunct in the lambda form of the specification. AMPHION's specification language for this domain is at the level of abstract Euclidean geometry (e.g., points, rays, ellipsoids, and intersections) augmented with astronomical terms (e.g., photons and planets). There is no mention of implementation level coordinate frames, units, and so on, except in defining representations for inputs and outputs. These are introduced during program synthesis.

Figure 2 is the diagram created interactively with AMPHION for this problem. AMPHION translated this diagram to the lambda form of the specification, and then the program synthesis subsystem generated the FORTRAN-77 program in Figure 3 in 96 seconds

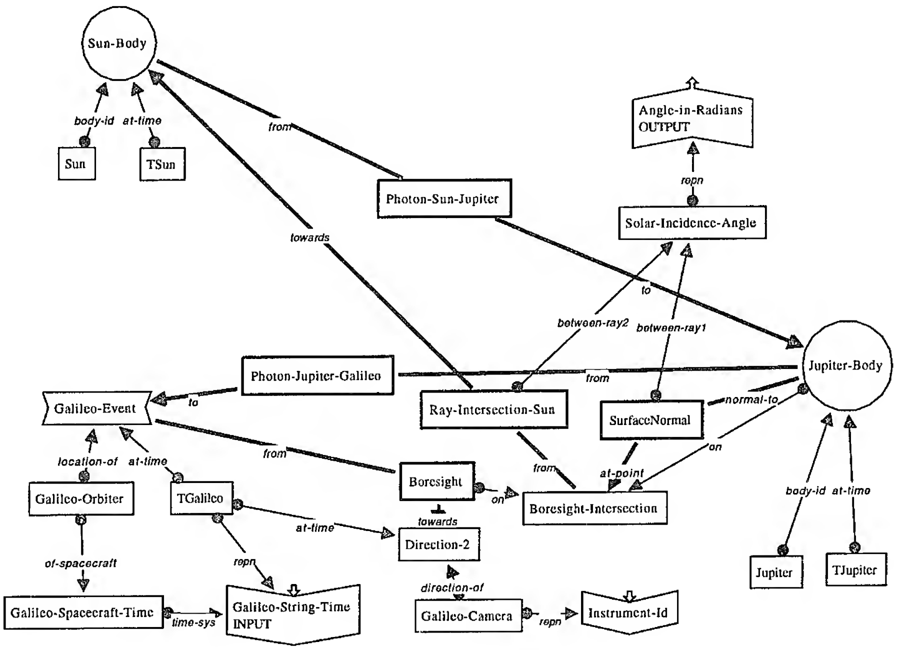


Figure 2: Diagram for solar incidence angle developed interactively with AMPHION.

```
SUBROUTINE SOLAR
        (INSTRU, GALILE, ANGLEI )
C      Input Parameters
INTEGER INSTRU
CHARACTER*(*) GALILE
C      Output Parameters
DOUBLE PRECISION ANGLEI
C      Function Declarations
DOUBLE PRECISION VSEP
C      Parameter Declarations
INTEGER JUPIT
PARAMETER (JUPIT = 599)
INTEGER GALIL
PARAMETER (GALIL = -77)
INTEGER SUN1
PARAMETER (SUN1 = 10)
DOUBLE PRECISION TIKTOL
PARAMETER (TIKTOL = 0.01)
C      Variable Declarations
Deleted for lack of space
C    Dummy Variable Declarations
INTEGER DMY10
DOUBLE PRECISION DMY20 ( 6 )
DOUBLE PRECISION DMY60
DOUBLE PRECISION DMY61
LOGICAL DMY62
DOUBLE PRECISION DMY80 ( 6 )
LOGICAL DMY170
```

```
CALL BODVAR ( JUPIT, 'RADII', DMY10, RADJUP )
CALL SCS2E ( GALIL, GALILE, E )
CALL SCE2T ( INSTRU, E, S )
CALL SPKSSB ( GALIL, E, 'J2000', PVGALI )
CALL SPKEZ ( JUPIT, E, 'J2000', 'NONE', GALIL,
                   DMY20, LTJUGA )
CALL CKGPAV ( INSTRU, S, TIKTOL, 'J2000', C,
                   DMY60, DMY61, DMY62 )
CALL VEQU ( PVGALI ( 1 ), V1 )
X = E - LTJUGA
CALL VEQU ( C ( 3, 1 ), V3 )
CALL SPKSSB ( JUPIT, X, 'J2000', PVJUPI )
CALL SPKEZ ( SUN1, X, 'J2000', 'NONE', JUPIT,
                   DMY80, LTSUJU )
CALL BODMAT ( JUPIT, X, MJUPIT )
CALL MXV ( MJUPIT, V3, XV3 )
CALL VEQU ( PVJUPI ( 1 ), V2 )
X1 = X - LTSUJU
CALL VSUB ( V1, V2, DV2V1 )
CALL SPKSSB ( SUN1, X1, 'J2000', PVSUN1 )
CALL MXV ( MJUPIT, DV2V1, XDV2V1 )
CALL VEQU ( PVSUN1 ( 1 ), V )
CALL SURFPT ( XDV2V1, XV3, RADJUP ( 1 ),
   RADJUP ( 2 ), RADJUP ( 3 ), P, DMY170 )
CALL SURFNM ( RADJUP ( 1 ), RADJUP ( 2 ),
   RADJUP ( 3 ), P, PP )
CALL VSUB ( P, V2, DV2P )
CALL MTXV ( MJUPIT, DV2P, XDV2P )
CALL VSUB ( V, XDV2P, DXDV2V )
CALL MXV ( MJUPIT, DXDV2V, XDXDV2 )
ANGLEI = VSEP ( XDXDV2, PP )
      RETURN
      END
```

Figure 3: SOLAR program generated by AMPHION from Figure 2.

of CPU time on a Sparc 2. The appearance of icons and edges in a diagram can be modified to a user's preference. The general convention for edges in Figure 2 is that they are directed from defining variables to defined variables; the label on an edge describes the role of a defining variable in defining a defined variable. The function for defining a variable in terms of other variables is not displayed, but can be readily inferred from the edges leading into a defined variable. Rays and photons are an exception to this convention for edge direction. Photons denote constraints for light-time correction.

4 Domain Engineering

An AMPHION domain theory encodes the knowledge needed to correctly use the subroutines in a library. AMPHION's purpose is to map a problem specification to the functionality embedded in a subroutine libıary; the resulting program then solves the problem. AMPHION does not synthesize or verify the subroutines themselves. An AMPHION domain theory consists of an abstract theory that provides the background knowledge and specification language for formulating problems, a concrete theory for formalizing the subroutines, and an implementation relation between the abstract and concrete theory. The style of axiomatization is similar to algebraic specifications for abstract data types: in essence abstract types and operations are created for the abstract theory, and then also for the concrete theory. The operations are then axiomatized, with particular emphasis on the implementation relation between abstract and concrete types and operations.

Developing the abstract theory is not a straightforward process of formalizing an existing ontology, rather it is a creative iterative process that, if done well, results in a conceptual framework that guides users in formulating their problems. An iterative refinement methodology was used for creating the abstract domain theory for solar system kinematics, similar to that espoused by Lakatos [4]. Starting with informal requirements for a set of nine typical problems in this domain, a domain expert and experts in KBSE concurrently developed the vocabulary and formal specifications for these problems. Several iterations were required over a period of a couple of weeks to develop a satisfactory vocabulary in which all the problems could be succinctly and naturally described.

The structure of an AMPHION domain theory will be illustrated in the following subsections using fragments of the domain theory needed to solve the Galileo example. The axioms have been simplified for purposes of presentation, mainly by including only one dimension of representational choice – coordinate frames – and omitting other dimensions of representational choice, such as time systems and coordinate systems.

4.1 Abstract Theory

The abstract domain theory includes types for objects in Euclidean geometry augmented with astronomical constructs such as photons, planets (modeled by ellipsoids), and spacecraft. Abstract types are independent of any particular representation. Abstract functions include constructors for derived types, such as the constructor for a ray from a point and a direction, and their corresponding selectors. Geometric operations are also included as abstract functions, such as intersecting one geometric object with another. The abstract relations include geometric predicates, such as whether one geometric object intersects another.

The semantics of functions and relations that correspond to concrete subroutines are defined by the implementation axioms. The subsection on the implementation relation below describes how the function *intersect-ray-ellipsoid*, which determines the point where a given ray first intersects an ellipsoid, is axiomatized. The semantics of the remaining functions and relations fall into two categories. First are those that are definitions based on other abstract functions and relations. For example, the equatorial plane of a planet is defined as the plane perpendicular to the north pole at the center of a planet. The second category are non-definitional axioms among abstract functions and relations. For example, the relation *lightlike?* between two bodies at two different times holds if a photon leaving the center of the first body at the earlier time would arrive at the center of the second body at the later time. Photons in a diagram are rewritten into *lightlike?* relations. The *lightlike?* relation is axiomatized in terms of two abstract functions, which in turn are axiomatized in the implementation relation: the function *a-received* returns the time a signal would be received on the *r-body* if it was sent from *s-body* at time *s-time*; the function *a-sent* is the inverse. The following axioms mutually define the *lightlike?* relation and the *a-received* and *a-sent* functions (all variables are universally quantified):

lightlike?(s-body,s-time, r-body, a-received(s-body,r-body, s-time))
lightlike?(s-body,a-sent(s-body, r-body, r-time), r-body, r-time)

Given a set of *lightlike?* relations translated from photons in a diagram, the theorem prover generates substitution terms consisting of applications of the *a-received* and *a-sent* functions for all the sending- and received-time logical variables. This is done through unification during a unit resolution of the (negated) specification and one of the axioms above. For the Galileo example in Figure 2, two such terms are generated (where *TGalileo* is declared to be an input to the program):

TJupiter ← *a-sent(Jupiter, Galileo-Orbiter, TGalileo)*
TSun ← *a-sent(Sun, Jupiter, TJupiter)*

These abstract terms are later transformed into subroutine calls involving light-time correction (e.g., the *spkez* SPICELIB subroutine), taking into account necessary time-system conversions, in a manner similar to the derivation described below.

4.2 Concrete Theory and Implementation Relation

The concrete theory defines types used in implementing a program. The Galileo example uses the type 3Vector, which is a vector of three reals that variously represent a spatial position, direction, or the lengths of the three radii of an ellipsoid. In general, there is a many-to-many correspondence between abstract types and concrete types. However, any particular instance of a concrete type represents only one abstract type, defined through an *abstraction map*. The implementation relation is axiomatized in the style of Hoare [3] through these abstraction maps from concrete types to abstract types. These abstraction maps are often parameterized. To facilitate posting constraints on abstraction maps, the abstraction maps are also reified. *Abs* is used to apply a reified abstraction map to a concrete object, e.g., *abs(coord-to-point(F), c)* denotes applying the abstraction map *coord-to-point*, parameterized on the coordinate frame *F*, to the 3Vector *c*. A frame is an origin and three perpendicular axes. *Coord-to-dir* is a similar abstraction map for directions. *Radii-to-ellipsoid* maps the lengths of three radii to an ellipsoid, given a coordinate frame assumed to be aligned along the ellipsoid's axes.

The functions of the concrete theory denote subroutines in the target subroutine library. The SPICELIB subroutine *surfpt,* given the coordinates of an observation point, the coordinates of an observation direction, and three radii of an ellipsoid assumed to be aligned along the coordinate frame; returns the coordinates where the observing vector first intersects the ellipsoid. The observation point and direction, as well as the intercept coordinate, must all be represented in the coordinate frame defined by the ellipsoid.

The following equation is a typical implementation axiom: it defines how the abstract function *intersect-ray-ellipsoid* – used in the Galileo example – is implemented by the target language function *surfpt:*

```
intersect-ray-ellipsoid (origin-and-direction-to-ray(
                                   abs(coord-to-point(F), oc),
                                   abs(coord-to-dir(F), dc)),
          abs(radii-to-ellipsoid(F), radii))
 = abs(coord-to-point(F), surfpt(oc,dc,radii))
```

This equation, like all the implementation axioms, has the structure of a commutative diagram: applying the abstract function to the abstraction of the concrete inputs yields the same result as abstracting the output of applying the concrete function to the concrete inputs. This axiom also expresses the constraint that the concrete *surfpt* function implements the abstract *intersect-ray-ellipsoid* function only when the ellipsoid frame, the observing frame, the direction frame, and the frame for the intersection point are all identical (e.g., *F*).

To use this axiom in the Galileo example, the theorem prover must introduce a coordinate conversion between the coordinate frame for the ellipsoid – *Jupiter-Frame* – and the coordinate frame for the *Boresight* ray. The latter is the standard *J2000* frame, a helio-centered frame whose z-axis points to the north star in the year 2000. The ephemerides for planets and spacecraft are represented in *J2000*. The introduction of this coordinate conversion is done through one of the axioms defining coordinate frame conversions (the other axioms define these conversions as a group of transformations):

```
abs(coord-to-point(f1),v) =  abs(coord-to-point(f2), coord-convert(f1,f2,v))
```

The following conjunct defines *Boresight-intersection* at an intermediate step in the program derivation:

```
Boresight-Intersection =
 intersect-ray-ellipsoid (origin-and-direction-to-ray(
                                  abs(coord-to-point(J2000), Galileo-Event),
                                  abs(coord-to-dir(J2000), Direction-2)),
          abs(radii-to-ellipsoid(Jupiter-Frame), Jupiter-radii))
```

Equality replacement (paramodulation) with the coordinate conversion axiom above and a similar one for direction conversion yields a transformed conjunct with two new logical variables for frames, *F1* and *F2*:

```
Boresight-Intersection =
   intersect-ray-ellipsoid (origin-and-direction-to-ray(
                   abs(coord-to-point(F1), coord-convert(J2000,F1,Galileo-Event)),
                   abs(coord-to-dir(F2), dir-convert(J2000,F2,Direction-2))),
          abs(radii-to-ellipsoid(Jupiter-Frame), Jupiter-radii))
```

The right hand side of this equation unifies with the left hand side of the *intersect-ray-ellipsoid* implementation equation, with the following substitutions:

F ← *Jupiter-Frame*   *F1* ← *Jupiter-Frame*   *F2* ← *Jupiter-Frame*

Then, again through paramodulation, the following concrete-level definition is generated for the *Boresight-Intersection* point:

```
Boresight-Intersection =
        abs(coord-to-point(Jupiter-Frame),
            surfpt-intercept(coord-convert(J2000,Jupiter-Frame,Galileo-Event),
                             dir-convert(J2000,Jupiter-Frame,Direction-2),
                             Jupiter-radii)
```

5 Specification Acquisition

AMPHION's GUI achieves the benefits associated with structured editors and visual programming paradigms - but at the specification level rather than the program level. Furthermore, the information needed to instantiate the GUI is derived from the declarative domain theory. This guarantees consistency between the specification acquisition subsystem and the program synthesis subsystem when a domain theory is updated. The GUI's cascading menus for adding and refining objects in a specification diagram incorporate the functionality of a structured editor by presenting a user with a template for defining/refining an object according to the syntax of the domain theory. The slots in these templates are themselves menus enumerating the possible choices, given the current state of the specification. A user can also directly manipulate the icons and edges in a specification diagram, as in a visual programming environment, with the GUI disallowing actions that would violate the domain theory. The net effect of these capabilities is that a user does not need to learn the syntax of a textual specification language or the terminology of formal logic to develop problem specifications with AMPHION.

The specification acquisition subsystem incorporates a number of additional mechanisms that greatly facilitate users' developing correct specifications. The GUI ensures that specifications are well-defined by enforcing a well-founded ordering based on one object being used in the definition of another object. The GUI has top-down, bottom-up, and selected-object(s) modalities that support different methodologies for developing well-defined specifications. A specification is also semantically checked before program synthesis by attempting to solve an abstracted version of the specification, obtained by deleting conjuncts defining concrete representations for program inputs and outputs. If a valid abstract program cannot be derived, then the user is given feedback identifying errors such as over-constrained and under-constrained variables.

A domain theory developer can also allow overloading of the abstract functions and relations used in developing a specification. This reduces clutter in a diagram, and more importantly enables a user to think about the semantics of a problem rather than syntactic issues of typing. In general, whenever it is obvious how one type can be coerced into another type; then overloaded types, functions, and relations are defined for use by the GUI. From a table of coercions, AMPHION creates an expanded domain theory for the GUI. Axioms are generated defining the types, functions, and relations in the expanded GUI theory in terms of the types, functions, and relations in the domain theory. These axioms enable AMPHION to translate specifications developed in the expanded GUI theory to the more restricted theory used for program synthesis. The expanded GUI domain theory abstracts away irrelevant syntactic detail for the user while still ensuring that specifications are typed correctly for program synthesis.

For example, a body (a planet or moon) can be viewed as the point defined by its center. In the expanded GUI theory, a new supertype is generated called *Coercible-to-point*, of which body is a subtype. Functions like *distance* are overloaded with these supertypes, and the axioms defining these overloaded functions are generated:

overloaded-distance(x,y) = distance(coerce-to-point(x), coerce-to-point(y))

The function *coerce-to-point* coerces its argument, whatever type it may be, into a point, according to the entries in the table of coercions. The action of *coerce-to-point* on bodies could be defined as follows:

isbody?(b) => coerce-to-point(b) = center-of(b)

However, in AMPHION this coercion axiom is reformulated as an unconditional equality which is used as a more efficient unconditional rewrite rule:

coerce-to-point(body(b)) = center-of(b)

In the axiom above, the function *body* is a *retract* [10] which serves the same role as the type predicate in the conditional equality.

6 Related Work and Summary

This work is most closely related to that of Tyugu [11] in which software is also composed from subroutine libraries, but in contrast to Tyugu uses full predicate logic instead of intuitionistic propositional logic. The formal approach to domain-specific software design environments taken in this work contrasts with the ad-hoc approach championed by Fisher [2] and previous approaches to domain-specific automatic programming [5].

This paper has described a methodology for creating specification-based programming environments for domains with mature subroutine libraries. Raising development, modification, and reuse to the specification level eliminates an inherent knowledge barrier to using even the best-engineered subroutine libraries. This methodology has been implemented in the AMPHION system, which includes generic specification acquisition and automatic program synthesis subsystems driven by a declarative domain theory. AMPHION is applied to a domain by developing a domain theory structured according to the methodology described in this paper.

Acknowledgments

Mark Stickel developed the SNARK FOL theorem prover and with Richard Waldinger adapted SNARK to deductive program synthesis. Waldinger aided reformulating a version of the domain theory adapted to a previous theorem proving platform to SNARK's notation, and collaborated in running initial test cases through SNARK. The anonymous reviewers, Jeffrey Van Baalen, and Linda Wills provided valuable advice for revising this paper.

References

1. C. Green, D. Luckham, R. Balzer, T. Cheathan, and C. Rich: Report on a Knowledge-Based Software Assistant, in C. Rich, R.C. Waters (eds.): Artificial Intelligence and Software Engineering. Los Altos: Morgan Kaufmann, 1986, 337-428.
2. G. Fischer: Domain-Oriented Design Environments, KBSE'92, 204–213.
3. C.A.R. Hoare: Proof of Correctness of Data Representations, Acta Informatica 1, 271-281.
4. E. Lakatos: Proofs and Refutations, J. Worrall (ed.), Cambridge University Press, 1976.
5. M. Lowry and R. McCartney (eds.): Automating Software Design, MIT Press 1991.
6. M. Lowry, A. Philpot, T. Pressburger, and I. Underwood: A Formal Approach to Domain-Oriented Softwre Design Environments, KBSE'94.
7. Z. Manna and R. Waldinger: Fundamentals of Deductive Program Synthesis, IEEE Transactions on Software Engineering (18) 8, August 1992, 674-704.
8. D.R. Smith: KIDS: A Semiautomatic Program Development System, IEEE Transactions on Software Engineering 16,9 (1990), 1024-1043.
9. M. Stickel, R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood: Deductive Composition of Astronomical Software from Subroutine Libraries, 1994, in CADE-12.
10. J. Stoy: Denotation Semantics: the Scott-Strachey approach to programming language semantics. MIT Press, 1977. *Retract defined on page 133.*
11. E.H. Tyugu, *Knowledge-Based Programming*, Turing Institute Press, Glasgow, Scotland, 1988.

RUTH : an ILP Theory Revision System

Hilde Adé, Bart Malfait and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium

Abstract. We present the system RUTH (Revising and Updating THeories) which represents an incremental ILP approach to theory revision. The approach integrates intensional database updating and incremental concept-learning. RUTH uses a set of operators in order to make a given knowledge base consistent w.r.t. a user input integrity theory. Important is that apart from adding and deleting clauses and facts, we also employ an abductive operator, which allows RUTH to introduce missing factual knowledge into the knowledge base. In order to guide the search, several heuristics are used, on top of an intelligent search strategy derived from iterative deepening.

keywords : *Theory Revision, Inductive Logic Programming*

1 Introduction

The system presented in this paper belongs both to the areas of Inductive Logic Programming (ILP) and Theory Revision. Inductive Logic Programming (see [11]) is a research area that recently emerged at the intersection of Logic Programming and Machine Learning. In general ILP learning systems are concerned with the derivation of logic programs from given background knowledge and positive and negative evidence for one or more predicates.

Theory Revision is a research area where one is concerned with the development of tools for updating knowledge bases while maintaining the overall consistency. A theory revision system is thus able to refine, restructure and correct the knowledge base, and to incrementally incorporate newly incoming information.

In [6] it is argued that both incremental concept-learning and intensional database updating (see [4, 7, 17]) are instances of the more general problem of belief updating when reformulating both problems in a logical framework. As proof for this claim, an adaptation and extension of Shapiro's Model Inference System was presented, which is able to solve the belief updating problem interactively, i.e., guiding its search by asking questions to the user.

The system RUTH we will discuss here is an implementation of the TR-system presented in [1] with some important enhancements. It solves the theory revision problem for a restricted subset of Prolog, i.e., the system is able to cope with definite, functor-free Horn clauses, with existentially quantified variables up to a certain level in the body. The system works non-interactively, using an intelligent search strategy derived from iterative deepening, enhanced with several heuristics. Moreover, it is shown in [1] that the system is sound and complete. That means, if the system outputs an updated knowledge base, it will be consistent with the integrity constraints. And, if there exists a solution for the update problem, the system will find one using the given set of operators : adding and deleting clauses and ground facts, and abduction.

This paper is organized as follows : in section 2 we give a formal specification of the problem. Section 3 describes our system RUTH declaratively and procedurally. Section 4 discusses experiments on the "Student Loan"-domain (see [12]). In section 5 we describe related approaches and finally we formulate conclusions in section 6.

2 Problem specification

The theory revision problem can be formalized as follows :

- **Given :**
  - an integrity constraint theory I
  - a new integrity constraint IC
  - a database D, satisfying I, but possibly not satisfying IC
  - a language $\mathcal{L}$, i.e., the specification of the syntactic bias that determines the set of well-formed clauses that are allowed in the database D
- **Find** : a transaction *Trans* such that database *Trans*(D) satisfies I ∪ {IC}, and *Trans*(D) is in the language $\mathcal{L}$

In our settings an *integrity theory* is a set of integrity constraints. An *integrity constraint* is a clause of the form $p_1 \wedge ... \wedge p_k \rightarrow q_1 \vee ... \vee q_m$, (k≥0, m≥0), which can be read as: if p_1 and ... and p_k are true, then at least one of the q_j should be true.

A *database* is a set of definite clauses, and as language $\mathcal{L}$ we chose functor-free Horn clauses, with the extra restriction that clauses with a non-empty body cannot contain constants. Finally, a *transaction* is a list of actions of the form *add(fact), add(clause), delete(fact)* or *delete(clause).* Our notion of a clause (constraint) being *satisfied* by a database, is the so-called *definite* semantics (see [11]), which exploits the fact that a definite clause theory D has a unique least Herbrand model $\mathcal{M}$(D).

Definition 1. A clause c *satisfies* a database D iff for all substitutions θ such that body(c)$\theta \subseteq \mathcal{M}$(D), there is at least one literal l_i in head(c) such that $l_i\theta \in \mathcal{M}$(D). A clause c *violates* a database D iff there exists a substitution θ such that body(c)$\theta \subseteq \mathcal{M}(\mathcal{D})$, but there is no literal $l_i \in$ head(c) such that $l_i\theta \in \mathcal{M}$(D).

3 RUTH

We describe our system RUTH that handles the theory revision problem as it is formalized in the previous section. We discuss the basic algorithm, present the different operators being used, and illustrate the system on a small example.

3.1 The basic algorithm

When a database D violates an integrity constraint $p_1 \wedge ... \wedge p_k \rightarrow q_1 \vee ... \vee q_m$ with answer substitution θ, then at least one of the $p_i\theta$ should be false, or at least one of the $q_j\theta$ should be true in the transformed database D_{new}. In predicate learning such a $q_j\theta$ is called an uncovered positive example for the predicate in the literal p_i. Analogously, $p_i\theta$ is called a covered negative example.

Therefore, upon violation of a constraint, the algorithm will hypothesize the classification of unknown examples as true or false, i.e., it will make assumptions as to which of the $p_i\theta$ or $q_j\theta$ are responsible for the violation of the constraint.

The following datastructures are used :

- D : contains the initial database.
- I : the set of constraints that should be satisfied by the database .
- Trans : contains the current transaction, i.e., the current list of modifications. Initially this list is empty. Note that Trans denotes the transaction, and *Trans*(D) denotes the transformed database after applying the operations in the list Trans on the initial database D.
- Hypo : the list of examples, whose classification was hypothesized for repairing the constraint violation. Hypothesized examples are added in front of this list. Initially this list is empty.

```
procedure RUTH( I , D )
begin
      Trans ← ∅
      Hypo ← ∅
      while I is still violated by Trans(D) do
           if Hypo ≠ ∅
           then
                choose an example Ex from Hypo                  ①
                if Ex is an uncovered positive example
                then handle_positive(Trans,I,Hypo,Ex)           ②
                else { Ex is a covered negative example }
                      handle_negative(Trans,I,Hypo,Ex)          ③
           else
                select a violated constraint Ic from I          ④
                hypothesize an example Ex                       ⑤
                Hypo ← Hypo ∪ Ex
Output : Trans(D)
end procedure RUTH
```

Fig. 1. The top-level algorithm of RUTH

Figure 1 illustrates the top-level algorithm. RUTH starts with an empty transaction Trans and an empty list Hypo. As long as the transformed database

*Trans(*D*)* violates the integrity theory I, it enters its while-loop. If there are still examples in the list Hypo that need to be handled, RUTH chooses one of them and treats it accordingly. Roughly speaking, the list Hypo is handled in a LIFO order. New examples are always added in front of Hypo, and by choosing the first example in the list to be treated, RUTH always handles the most recent problem. Note that the procedures *handle_positive* and *handle_negative* always change the datastructures Trans or Hypo. When there are no examples in Hypo, then RUTH selects a violated constraint. Suppose this constraint is of the form $p_1 \wedge ... \wedge p_k \rightarrow q_1 \vee ... \vee q_m$, and that it is violated with substitution θ. This means that in order to repair this constraint violation, either one of the $p_i\theta$ should be false, or one of the $q_j\theta$ should be true in $\mathcal{M}$(D). The system therefore selects such a $p_i\theta$ or $q_j\theta$ and adds it to Hypo together with the appropriate classification (*false* or *true*). Note that in this way we avoid asking questions to an oracle, as in interactive theory revision systems.

When all integrity constraints are satisfied by *Trans(*D*)*, the system stops, and outputs the transformed knowledge base.

3.2 The operators of RUTH

Handling positive examples. To handle uncovered positive examples, RUTH has three operators :

1. add the example Ex as a fact to the database D. This is not always an elegant solution, but sometimes the system just needs to add factual knowledge to the database.
2. build a maximally general clause within the language of RUTH, that covers Ex, and that is consistent with the negative examples in Hypo, and the constraints in I that denote negative examples. For this operator we use the generalization procedure of Shapiro's MIS system (see [15]), but with an adapted refinement operator that reflects the language of RUTH.
3. abduce one or more new facts. For this a clause $h \leftarrow l_1 \wedge \ldots \wedge l_n$ of the database is chosen, such that h=Exθ and for which not all literals $l_i\theta$ are true. These literals are then added as positive uncovered examples to Hypo.

As a default, all three of these operators may be invoked for all predicates in the knowledge base. However, we provide the user with the possibility to restrict the allowed operations on each of the predicates.

Note that when there are existentially quantified variables in the body of a clause, one should be careful when applying abduction. The system should take care that all arguments in the abduced literals are instantiated, before adding them to Hypo. In RUTH this is done using types and domains. With each argument place in each predicate, a type is associated, and with each type a domain is associated, i.e., a set of constants that are of the given type. When RUTH abduces a literal where one of the arguments is not instantiated, it looks for the associated type, and fills in a constant from the domain belonging to that type. Upon backtracking the other elements in the domain will be tried.

Handling negative examples. Handling a negative covered example Ex is done as in Shapiro's specialization procedure. By analyzing the SLDNF-tree for a covered negative example, Shapiro's procedure locates an incorrect clause by asking questions to the user. Suppose this clause has the form $h \leftarrow l_1, \ldots, l_n$, and θ is the substitution applied to it in the proof for Ex. In our approach, which does not generate questions, we would hypothesize that one of the clauses that contributes to the proof of Ex is incorrect. This corresponds to hypothesizing that there is substitution θ for which the head of a clause is false, and its body is true. The system then retracts this clause from the database, adds $h\theta$ as a negative example to Hypo and adds all $l_i\theta$ as positive examples to Hypo.

3.3 A small example

Suppose the following database is given :
1 $good\_mood(John) \leftarrow$
2 $smiles(John) \leftarrow$
3 $happy(John) \leftarrow$
4 $good\_mood(Anna) \leftarrow$
5 $smiles(Anna) \leftarrow$
6 $is\_married(Anna) \leftarrow$
7 $happy(X) \leftarrow good\_mood(X) \wedge smiles(X)$
8 $happy(X) \leftarrow has\_friends(X)$
Suppose the integrity theory $\mathrm{I} = \{happy(John) \rightarrow\ ,\ is\_married(X) \rightarrow happy(X)\}$ is supplied to the system, and suppose the system decides to handle $happy(John) \rightarrow$ first. This constraint is violated, since the query *?- happy(John)* succeeds. The system then hypothesizes *happy(John)* as a negative covered example. This example is then added to Hypo together with the classification *false*. While handling this example, the system builds the proof tree shown in Figure 2.

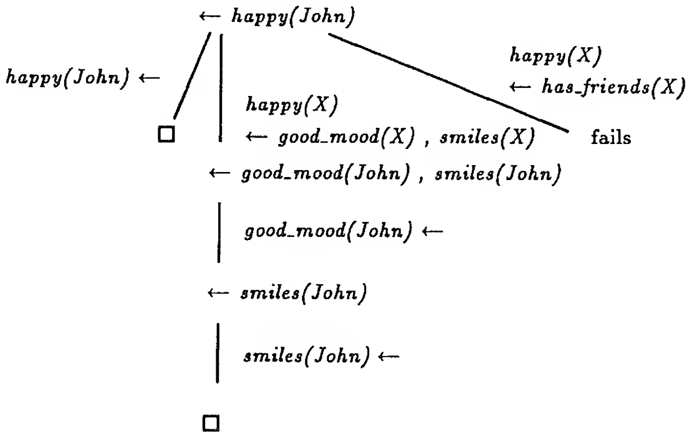


Fig. 2. The proof tree for *happy(John)*

First the algorithm can choose among deleting 3 or 7. Suppose we take the choice point where clause 3 is deleted, i.e., RUTH enters its while-loop with Trans = {*del(3)*} and Hypo = {*(happy(john), false)*}. Since Trans(D) still does not classify the example *happy(John)* as false, it is handled again. The proof tree for the example is constructed (this proof tree is identical to the old one except that the left branch containing *happy(John)* is deleted). To disable this proof, we can cut the first branch of the proof tree in three ways, i.e., by deleting clause 1, 2 or 7. Either one of these is sufficient.

Since now all examples in Hypo are satisfied, the system will again check for violation of constraints in I. It finds out that the constraint *is_married(X)* → *happy(X)* is violated for the substitution θ = {X/Anna}. So either *married(Anna)* is a negative covered example, or *happy(Anna)* is a positive uncovered example. Suppose the system chooses to hypothesize *happy(Anna)* as a positive example. For handling this example, there are again several possibilities:

1. Extend Trans with *add(happy(Anna)←)*.
2. Use the clause *happy(X)* ← *has_friends(X)* for abduction, i.e., hypothesize *has_friends(Anna)* as a positive example (and add it as such to Hypo).
3. Construct a clause that covers *happy(Anna)*, and that does not cover *happy(John)*. The procedure *missing-clause* will produce the clause *happy(X)* ← *is_married(X)*. This is the shortest clause that covers *happy(Anna)* and that does not cover *happy(John)*. Therefore the system will add *add(happy(X)* ← *is_married(X))* to Trans.

If option 3 is chosen, Trans(D) satisfies all examples in Hypo. The system then checks whether Trans(D) satisfies I. Since it does, the procedure halts, and outputs Trans(D), with Trans = {*add(happy(X)* ← *is_married(X)), delete(happy(X)* ← *good_mood(X)* ∧ *smiles(X))* , *delete(happy(John))* }.

3.4 Search strategy

In this section we discuss the search strategy being applied in RUTH. Note that we can basically use any complete search strategy. However, because of the large number of alternatives, and since the memory requirements for storing one such alternative is high, neither depth-first nor breadth-first search seem appropriate. Therefore we chose a depth-first iterative deepening scheme (see [8]), with the depth of a solution being defined as the length of a transaction, i.e., the number of operations in a transaction.

The general heuristic incorporated in the system is to explore the more promising solutions earlier in the search process. Therefore the shortest transactions, and those with the least hypothesized examples in the list Hypo will be explored first. This heuristic also expresses a *minimality criterion*, in that it prefers solutions with the least changes to the knowledge base.

We also try to employ as much as possible user input knowledge that can significantly reduce the search space. For each predicate in the knowledge base the user can (but need not) restrict the possible operator applications. To this purpose three meta-predicates are defined : base/1 (which means that only facts can be added), view/1 (which means that clauses can be built) and abducible/1.

The user has also the possibility to formulate a bias that restricts the refinement operator for building clauses. This is done by providing the system with clause models formulated in the so-called Extended Inductive Programming Language (see [2]).

The general search heuristic (i.e., preferring the shortest transaction first) is already reflected in the choice for an iterative deepening scheme as search strategy. However, several other heuristics are built on top of this strategy. These heuristics guide the search, at the points where the algorithm has several choices (see the labels ①, ②, ③, ④ and ⑤ in the algorithm of Figure 1). A full discussion of the heuristics is beyond the scope of this paper. For more information we refer to [3] and [10].

Finally, we built in pruning heuristics that make the system avoid redundancies in transactions. E.g., we do not allow that RUTH first deletes a clause, and then adds the same clause again to the database.

Important to note here is that the set of operators we defined, is complete[1], and that the search strategy and the heuristics we use, guide the system through the search space, but do not prevent it from finding a solution when there exists one within the given settings. Indeed, whenever needed, the system will backtrack and eventually try all possibilities.

4 Experiments

In this section we describe and discuss the test results of applying RUTH to the "Student-Loan" domain, described in [12]. In the tests RUTH is given a rule base of a small expert system designed to indicate when a student is required to pay back an educational loan.

Figure 3 contains the original rule base. In order to apply RUTH to it, we intentionally introduced errors, by either adding or deleting clauses or literals in clauses. In addition RUTH is given a set of 24 positive and 24 negative randomly selected examples of the predicate *no_payment_due/1*, that indicates whether or not a student is required to pay back his or her educational loan. These examples are used as an integrity theory. Finally also the basic knowledge concerning the students is available. This knowledge consists of a set of ground facts for the predicates male/1, school/1, armed_forces/1, enrolled/3, longest_absence_from_school/2, filed_for_bankrupcy/1, armed_forces/1, peace_corps/1 and disabled/1.

We give here one example of a test result. More results can be found in [3] and [10]. The table lists the errors introduced in the knowledge base, the solution proposed by RUTH, the time[2] needed for finding the proposed solution, and the depth at which the solution is found.

[1] Completeness here means that if there is a solution, the system will find one.

[2] Times are given in cpu-seconds on a Sparc Sun4. The program RUTH is written in ProLog by BIM.

```
1  no_payment_due( Student ) :- continuously_enrolled( Student ) .
2  no_payment_due( Student ) :- eligible_for_deferment( Student ).
3  continuously_enrolled( Student ) :- never_left_school( Student ) ,
                                       enrolled_in_more_than_n_units( Student , N ) ,
                                       five( N ) .
4  never_left_school( Student ) :- longest_absence_from_school( Student , Absence ),
                                   greater_than( Absence , 6 ) .
5  enrolled_in_more_than_n_units( Student , N ) :- enrolled( Student , School , Units ) ,
                                                   school( School ) ,
                                                   greater_than( Units , N ).
6  eligible_for_deferment( Student ) :- eligible_for_military_deferment( Student ).
7  eligible_for_deferment( Student ) :- eligible_for_peace_corps_deferment( Student ).
8  eligible_for_deferment( Student ) :- eligible_for_financial_deferment( Student ).
9  eligible_for_deferment( Student ) :- eligible_for_student_deferment( Student ).
10 eligible_for_deferment( Student ) :- eligible_for_disability_deferment( Student ).
11 eligible_for_military_deferment( Student ) :- enlist( Student , Org ),
                                                 armed_forces( Org ).
12 eligible_for_peace_corps_deferment( Student ) :- enlist( Student , Org ) ,
                                                    peace_corps( Org ).
13 eligible_for_financial_deferment( Student ) :- filed_for_bankrupcy( Student ).
14 eligible_for_financial_deferment( Student ) :- unemployed( Student ) .
15 eligible_for_student_deferment( Student ) :- enrolled_in_more_than_n_units( Student , N )
                                                eleven( N ).
16 eligible_for_disability_deferment( student ) :- disabled( Student ).
```

Fig. 3. Student-Loan Rule Base

Errors	delete *enrolled(student775 , ucla , 6)* delete *clause 14* add *disabled(student89)*
Solution	[*delete(disabled(student89)),* *add(unemployed(student775)),* *add(no_payment_due(X) :- unemployed(X))*]
Time	4.91
Depth	3

This test illustrates that the solution found by RUTH is not necessarily the inverse of the errors that were introduced in the knowledge base. It also illustrates that once a clause is introduced in the knowledge base (here : *no_payment_due(X) :- unemployed(X).*), it can be used for abduction. RUTH uses this clause here to hypothesize *unemployed(student775)* as a positive example.

5 Related work

From the Logic Programming point of view, our work is related to [4] and [7]. However, as in the system described in [6], the extra possibility of adding clauses to the knowledge base is provided.

From the point of view of predicate learning, our work is related to [5], [14] and [15]. Our approach supports and extends the approach of [6] : Shapiro's algorithms are used and extended to handle integrity constraints. The main novelty is that we avoid the reliance on an oracle by using an intelligent search strategy.

The work is also related to the theory revisors AUDREY [18], FORTE [13] and RX [16]. The main difference between our approach and theirs is that we guarantee completeness, and that we employ general constraints instead of specific examples. These differences are significant. On the other hand, AUDREY, FORTE and RX are more efficient since they are more empirically oriented and employ heuristics.

Finally we mention KR-FOCL (see [12]). In [12] it is described how the system FOCL can be used as a knowledge revision tool. The difference with our system is that KR-FOCL works interactively (each update is proposed to the user, which then accepts or rejects it), and that it works in two passes. First a learning phase and then a revision phase, where based on the operations in the learning phase, and employing four heuristics, possible updates are formulated.

6 Conclusions

Working within the logical framework provided by the ILP research area, we have developed the theory revision system RUTH that updates logical theories w.r.t. newly incoming information. The system implements and enhances ideas presented in [1]. As opposed to [6] and systems such as KR-FOCL [12] our system works non-interactively : questions to the user are avoided by using an intelligent search strategy derived from iterative deepening, enhanced with heuristics that guide the system through the search space. Both the iterative deepening scheme and the heuristics reflect the general idea of preferring the shortest solution (minimality criterion). RUTH integrates techniques from the domains of machine learning and intensional knowledge base updating. On the one hand, it accepts evidence in the form of integrity constraints which are a generalization of the factual evidence (ground positive and negative examples) employed in the majority of predicate-learning systems. On the other hand it borrows from the machine learning domain a clause construction operator. As a result RUTH has at its disposal an elegant set of operators where both ground facts and clauses can be added or deleted to or from the database. In addition this set of operators is enhanced with an abductive procedure that allows the inference of missing factual knowledge. Moreover, the approach is complete for completely bound Horn clauses, and in principle for any finite extension of this representation language.

Acknowledgements

Research for this paper is funded by the ESPRIT BRA project Nr.6020 (Inductive Logic Programming), and co-financed by the Flemish Government under contract Nr.93/014. Luc De Raedt is supported by the Belgian National Fund for Scientific Research. We wish to thank Maurice Bruynooghe and Gunther Sablon for their comments on earlier versions of this paper.

References

1. H. Adé, L. De Raedt, and M. Bruynooghe. Theory revision. In *Proceedings of the 3rd International Workshop on Inductive Logic Programming*, 1993.
2. H. Adé, L. De Raedt, and M. Bruynooghe. Declarative Bias for Specific-To-General ILP Systems. To appear in *Machine Learning*, 1994.
3. H. Adé, B. Malfait, and L. De Raedt. Ruth : an ILP Theory Revision System. Technical Report CW-194, Department of Computer Science, Katholieke Universiteit Leuven, 1994.
4. Francois Bry. Intensional updates: abduction via deduction. In D. Warren and P. Szeredi, editors, *Proceedings of the 7th International Conference on Logic Programming*, pages 561–578. The MIT Press, 1990.
5. L. De Raedt. *Interactive Theory Revision: an Inductive Logic Programming Approach.* Academic Press, 1992.
6. L. De Raedt and M. Bruynooghe. Belief updating from integrity constraints and queries. *Artificial Intelligence*, 53:291–307, 1992.
7. A. Guessoum and J.W. Lloyd. Updating knowledge bases. *New Generation Computing*, 8:71–88, 1990.
8. R. Korf. Depth-first iterative deepening : an optimal admissable search. *Artificial Intelligence*, pages 97–109, 1985.
9. J.W. Lloyd. *Foundations of logic programming.* Springer-Verlag, 2nd edition, 1987.
10. B. Malfait. Een incrementele, niet-interactieve aanpak van het Theory Revision-probleem binnen ILP. Master's thesis, Department of Computer Science, Katholieke Universiteit Leuven, 1994. in Dutch.
11. S. Muggleton and L. De Raedt. Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 1994. To appear.
12. M. Pazzani and C. Brunk. Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition*, 3:157–173, 1991.
13. B.L. Richards and R.J. Mooney. First order theory revision. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 447–451. Morgan Kaufmann, 1991.
14. C. Sammut and R. Banerji. Learning concepts by asking questions. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume 2, pages 167–192. Morgan Kaufmann, 1986.
15. E.Y. Shapiro. *Algorithmic Program Debugging.* The MIT Press, 1983.
16. S. Tankitvanitch and M. Shimura. Refining a relational theory with multiple faults in the concept and subconcepts. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 436–444. Morgan Kaufmann, 1992.
17. A. Tomasic. View update translation via deduction and annotation. In *Proceedings 2nd International Conference on Database Theory*, volume 326 of *Lecture Notes in Computer Science*, pages 338–351. Springer-Verlag, 1988.
18. J. Wogulis. Revising relational theories. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 462–466. Morgan Kaufmann, 1991.

Dynamic Reducts as a Tool for Extracting Laws from Decisions Tables

Jan G. Bazan[1] Andrzej Skowron[2] and Piotr Synak[2]

[1] Institute of Mathematics, Pedagogical University
Rejtana 16A, 35-310 Rzeszow, Poland
e-mail: bazan@plumcs11.umcs.lublin.pl
[2] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
e-mails: skowron@mimuw.edu.pl
synak@mimuw.edu.pl

Abstract. We apply rough set methods and boolean reasoning for knowledge discovery from decision tables. It is not always possible to extract general laws from experimental data by computing first all reducts [12] of a decision table and next decision rules on the basis of these reducts. We investigate a problem how information about the reduct set changes in a random sampling process of a given decision table could be used to generate these laws. The reducts stable in the process of decision table sampling are called dynamic reducts. Dynamic reducts define the set of attributes called the dynamic core. This is the set of attributes included in all dynamic reducts. The set of decision rules can be computed from the dynamic core or from the best dynamic reducts. We report the results of experiments with different data sets, e.g. market data, medical data, textures and handwritten digits. The results are showing that dynamic reducts can help to extract laws from decision tables.

Key words: evolutionary computation, knowledge discovery, rough sets, decision algorithms, machine learning.

1 Introduction

The aim of the paper is to present a method for extracting laws from experimental data. Problems related to extracting laws from experimental data are intensively studied (see e.g. [5], [6,7], [8], [10] [16], [18], [1]). Our method is based on rough set approach [12] and boolean reasoning [2]. The rough set methods developed so far are not always sufficient for extracting laws from decision tables. The set of all decision rules generated from all conditional attributes can be too large or/and can contain many chaotic rules not appropriate for unseen object classification. The main problem can be formulated as follows: How to exctract a subset with bounded size from the set of all decision rules with as high as possible classification quality on unseen objects?
We propose to generate the decision rules on the basis of so called dynamic reducts. These reducts are in some sense the most stable reducts of a given decision table, i.e. they are the most frequently appearing reducts in subtables

created by random samples of a given decision table. The set of decision rules can be computed from dynamic reducts in two different ways.

One can choose the best dynamic reducts and compute the decision rules using only attributes from the dynamic core i.e. from the union of these dynamic reducts.

Another possibility is to compute the set of rules separately for any of the chosen dynamic reducts and to create the union of the constructed decision rule sets.

Decision rules are generated by applying boolean reasoning methods presented in [11], [19], [20].

New objects are matched against the computed decision rules. The final decision is predicted by applying some strategies to the decision set determined by the matched decision rules. One of the simplest strategy we used is the majority voting strategy.

Our method has been tested on several decision tables in particular on market data [3], textures [22], handwritten digits [24], medical data [23]. The results are suggesting that methods using dynamic reducts can be treated as a promising tool for extracting laws from experimental data.

2 Dynamic reducts

We assume that the reader is familiar with basic notions of the rough set theory [12] and methods of decision rules generation based on boolean reasoning [11], [20]. In particular by $\mathbb{A} = (U, A \cup \{d\})$ we denote the decision table and by $RED(\mathbb{A}, d)$ the set of relative reducts i.e. (in the case of consistent decision tables) the family of all minimal conditional attribute sets sufficient to discern between objects from different decision classes. The methods based on calculation of all reducts allow to compute, for a given consistent decision table $\mathbb{A}$, the descriptions of all decision classes of A in the form of decision rules exact in $\mathbb{A}$ that are complete and minimal with respect to descriptors (see [11], [20]). Unfortunately, the decision rules calculated in this way can be not appropriate to classify of unseen cases. We suggest the rules calculated by means of dynamic reducts are better predisposed to classify unseen cases, because they are the most stable reducts in the process of random sampling of the original decision table.

If $\mathbb{A} = (U, A \cup \{d\})$ is a decision table then any system $\mathbb{B} = (U', A \cup \{d\})$ such that $U' \subseteq U$ is called a subtable of $\mathbb{A}$.

Let $\mathbb{A} = (U, A \cup \{d\})$ be a decision table and $\mathbb{F}$ be a family of subtables of $\mathbb{A}$. By $DR(\mathbb{A}, \mathbb{F})$ we denote the set

$$RED(\mathbb{A}, d) \cap \bigcap_{\mathbb{B} \in \mathbb{F}} RED(\mathbb{B}, d)$$

Any element of $DR(\mathbb{A}, \mathbb{F})$ is called an $\mathbb{F}$-*dynamic reduct of* A.

From the definition of dynamic reducts it follows that a relative reduct of A is dynamic if it is also a reduct of all subtables from a given family $\mathbb{F}$. This notion can be sometimes too much restrictive so we apply also a more general notion of dynamic reducts. They are called $(\mathbb{F}, \varepsilon)$-*dynamic reducts*, where $\varepsilon \geq 0$. The set $DR_\varepsilon(\mathbb{A}, \mathbb{F})$ of all $(\mathbb{F}, \varepsilon)$-*dynamic reducts* is defined by

$$DR_\varepsilon(\mathbb{A}, \mathbb{F}) = \left\{ C \in RED(\mathbb{A}, d) : \frac{|\{\mathbb{B} \in \mathbb{F} : C \in RED(\mathbb{B}, d)\}|}{|\mathbb{F}|} \geq 1 - \varepsilon \right\}$$

Proposition 1.

1. *If* $\mathbb{F} = \{\mathbb{A}\}$ *then* $DR(\mathbb{A}, \mathbb{F}) = RED(\mathbb{A}, d)$.
2. *For any two families* $\mathbb{F}, \mathbb{F}'$ *of subtables of* $\mathbb{A}$, *if* $\mathbb{F} \subseteq \mathbb{F}'$ *then* $DR_\varepsilon(\mathbb{A}, \mathbb{F}') \subseteq DR_\varepsilon(\mathbb{A}, \mathbb{F})$.
3. *If* $\varepsilon \leq \varepsilon'$ *then* $DR_\varepsilon(\mathbb{A}, \mathbb{F}) \subseteq DR_{\varepsilon'}(\mathbb{A}, \mathbb{F})$.
4. $DR_\varepsilon(\mathbb{A}, \mathbb{F}) \supseteq DR(\mathbb{A}, \mathbb{F})$, *for any* $\varepsilon \geq 0$.
5. $DR_0(\mathbb{A}, \mathbb{F}) = DR(\mathbb{A}, \mathbb{F})$.

If $C \in RED(\mathbb{B}, d)$ *then the number* $1 - \inf\{\varepsilon \geq 0 : C \in DR_\varepsilon(\mathbb{A}, \mathbb{F})\}$ *is called the stability coefficient of* C *(relatively to* $\mathbb{F}$*).*

We have implemented two techniques for sampling of decision table and dynamic reduct computation.

The first is based on calculating of reduct traces. It consists of the following steps:

Step 1: Calculate reducts for a given information system.
Step 2: Delete randomly one (or more) row from the system.
Step 3: Calculate reducts for the received table.
Step 4: If the received table contains more than $40\% \cdot N$ of objects (where N is the number of objects in the decision table) then go to Step 2, else go to Step 5.
Step 5: For each reduct calculated in Step 1 take the number of sets calculated in Step 3 containing this reduct. This number is the *trace* of a given reduct.

Of course we are interested in reducts having the maximal traces (stable reducts), because such reducts are intuitively well predisposed to designate new examples. These reducts are taken as dynamic reducts.

The second method of decision table sampling and calculating dynamic reducts consists of the following steps. In the first step some samples of the decision table are computed.
For example:

10 samples with the size 90% of the decision table,
10 samples with the size 80% of the decision table,
10 samples with the size 70% of the decision table,
10 samples with the size 60% of the decision table,
10 samples with the size 50% of the decision table.

Thus we received 50 new decision tables. Next, reducts for all these tables are calculated. In the second step reducts with the stability coefficients higher than a fixed threshold are extracted. The reducts distinguished in such a way are treated as the true dynamic reducts.

3 Decision rules computed from dynamic reducts

If a set of dynamic reducts (with the stability coefficients greater than a given threshold) has been computed then it is necessary to decide how to compute the set of decision rules.

We have implemented two methods. The first one is based on *the* (F, ε)*-dynamic core* of A, i.e. on the set $\bigcup DR_{\varepsilon}(\mathbb{A}, \mathbb{F})$. We apply the methods based on boolean reasoning presented in [11], [19] to generate decision rules (with minimal number of descriptors) from conditional attributes belonging to the dynamic core. The second one is based on the decision rule set construction for any chosen dynamic reduct. The final decision rule set is equal to the union of all these sets.

In our experiments we have received slightly better results of tests when applying the second method.

If an unseen object has to be classified it is first matched against all decision rules from the constructed decision rule set. Next the final decision is predicted by applying some strategy predicting the final decision from all "votes" of decision rules. The simplest strategy we have tested was the majority voting i.e. the final decision is the one supported by the majority of decision rules.

4 Experiments with data

The experiments have been performed on following data: market data, handwritten digits, texture recognition, medical data. The computers, we used for experiments were HP Workstations (series 9000, model 735/99MHz, 42 MFlops, 80 SPECint92, 124 MIPS).

4.1 Description of experimental data

Market Data

We received from Hughes Research Laboratories [3] a package of financial data. There are five data sets. Each contains information on macro-economic market indicators and stock price changes in one of the five major international stock markets (New York, Tokyo, London, Paris, Frankfurt). There are 15 market indicators used, including industrial production, yield curve, money supply and inflation. Each of the indicators is characterised by four measures: **level**, **duration**, **short-trend** and **medium-trend**. These measures reflect the value of the indicator and a recent history of its changes. **Level** represents the value of the indicator, **duration** indicates the amount of time that the indicator has been at that value or close to it, **short-trend** and **medium-trend** denote the change of the value of the indicator (i.e. derivative) in the short term (i.e. one or two months) and medium term (i.e. six to twelve months). The values of measures and the value of stock price changes are given in a symbolic form derived from the numerical values.

The following symbolic values are used:

- for **level**: *very-low, low, normal, high, very-high* ....(values 1-5),
- for **duration**: *short, medium, long* ....(values 1-3),
- for **short-trend**: *down-strong, down, flat, up, up-strong* ......(values 1-5),
- for **medium-trend**: *down, volatile, flat, up* ....(values 1-4),
- for stock **price change**: *down-strong, down, flat, up, up-strong*..(values 1-5).

The problem is how to deduce the rules that map the financial indicators' data at the end of a given month into stock price changes a month later. It is assumed that the indicator information provided by a single line of data is sufficient to make that prediction. Thus it is not necessary to know the chronological order of the data lines.

Handwritten digits

The database used in this experiment is a part of the huge handwritten character database maintained by the National Institute of Standards and Technologies of the United States. It is called "NIST Special Database" and serves as a valuable source of data for many researchers all over the world. The data have been collected with scanners and different pointing devices to extract characters and digits from the special forms written by many people in the US. All digits in the database have one normalised format of 32×32 and are black-white. Information about a digit is stored in a file containing its binary image, which has the size of 128 bytes and its character identification (i.e. what type is the digit stored in the file). There are 1200 files of digits from 0 to 9 written by 18 different persons.
The main problem is a correct recognition of digits for which it has been designed to deal with. We have achieved a decision system with properly classified new examples.
We used a database of 700 digits for learning purposes and a set of 500 other digits for testing.
At the beginning, we have created a decision table based from the database. The attributes, used to create decision table, are of three different kinds: Mask Attributes, Modal Formula Attributes and Topological Attributes. The detailed description of them is in [24]. Then we have done our experiments with this decision table.

Texture recognition

We received some texture data [22] from Professor R. Swiniarski (Department of Mathematical Sciences, San Diego State University, US). The table consists of 24 condition attributes, one decision attribute and 320 objects. The conditional attributes were received by applying the Singular Value Decomposition method and a discretization method [22]. The decision attribute classifies the sample to one of ten textures: Tree bark, Brick wall, Plastic bubble, Calf leather, Knit, Raffia, Sand, Herringbone weave, Wood grain, Wool. The problem is to classify unseen texture to one of 10 (given above) textures. The training set of this table consist of 60% of objects and the testing set of the rest of the table.

Medical data

The Medical Database used in our experiment we have received from Dr S. Tsumoto (Tokyo Medical and Dental University, Japan) [23]. This database is originally prepared by him, during his work at the department of neurology in order to study the statistical characteristics of meningitis and the factors determining the prognosis of these diseases.
The database contains: 99 objects with 25 attributes.

4.2 Complexity problems and some approximation solutions

The processes of decision rules' generation from decision tables are based on the reduct set computation. The time of reduct set computation can be too long when the decision table has too many: attributes or different values of attributes or objects. The reason is that in general the size of the reduct set can be exponential with respect to the size of the decision table and the problem of computing of a minimal reduct is NP-hard [19]. Therefore we were forced to apply some approximation algorithms to obtain some knowledge about reduct sets. We can pursue the following courses of action: conceptual clustering values of attributes or groups of attributes; conceptual clustering of objects; synthesis of new attributes from decision table and joining of attributes into groups [1].

4.3 Results of experiments

We randomly divided any of considered decision table into two sets: the *training table* containing about 70% or 60% of all objects in original table, and the *test table* - the rest of the table. By *quality of classification* we mean the percent of objects in the test table for which the proper value of decision has been properly predicted by our decision algorithm. The experiments were repeated several times for different partitions of the original decision table into the training and testing parts.

Market Data

In our experiment [1] the decision table was divided into three sets of objects: $\mathbb{A}$ (with 30% of objects), $\mathbb{B}$ (with 30% of objects), $\mathbb{C}$ (with 40% of objects).
First the dynamic reducts of $\mathbb{C}$ have been calculated with the decision rule set for any dynamic reduct. From the obtained set of rules those rules have been chosen which were correct for the set $\mathbb{B}$. Finally the received rules have been tested on the set $\mathbb{A}$ in the following way: Any object from the set $\mathbb{A}$ has been matched against the decision rules and two successive values of decision d_1, d_2 (i.e. *up* and *up-strong*) have been computed. Let p_1, p_2 describe frequency of occurrences of d_1 and d_2.
The results of the experiment are the following:

1. for 44% of objects from $\mathbb{A}$ the decision was predicted properly, it means that for 44% of objects:

$$(p_1 > p_2 \text{ and } d_1 = d_0) \quad \text{or} \quad (p_2 > p_1 \text{ and } d_2 = d_0)$$

2. for 60% of objects from $\mathbb{A}$ the decision was predicted almost properly:

$$|p_1 - p_2| <= 0.3 \quad \text{and} \quad (d_1 = d_0 \text{ or } d_2 = d_0)$$

where d_0 is the correct value of decision.

The system has not predicted 13 cases (out of 58 in the test table) for which the final decision proposed was really poor, i.e. such that the absolute value of the difference

proper decision-final decision

was more than one.

The obtained quality of classification is much better than the quality received by applying other methods, e.g. the quality of classification received by ID3 (see [15]) was about 20%.

Although the obtained results may seem to be only satisfactory, we should stress that we find them very promising as the minor effects mainly result from poor quality of the available data, such as:

- market data tables contained some undefined (unmeasured: *no data*) values,
- the number of examples with extreme decisions (up-strong, down-strong) is too small,
- the knowledge in these data is too diffused.

Handwritten digits

We have done some experiments with the database of handwritten digits. The training set consisted of 700 objects with 80 attributes, of which 18 are topological, 46 are masks and 16 are modal logic attributes. The classification rate for training set was 100%, i.e. all 700 training objects were properly recognised. The method has been realised in two attempts. In the first attempt we joined attributes by 5 in each group, thus creating a decision table with 18 attributes. For that table we calculated all reducts and used reduct majority voting to classify new objects. By this method a recognition rate about 90% has been achieved. In the second attempt attributes have been grouped by 2 resulting in a table with 40 attributes. From this attribute set 83 reducts have been extracted. Three dynamic reducts have been found, each consisting of 26 grouped attributes. Rules produced by these reducts allowed to achieve for a new object recognition rate also about 90% (comparable with the results gained by all rules). Moreover, rules computed from one of these three reducts allowed to achieve, almost the same, the new object recognition rate. This experiment is showing that dynamic reducts can be used as an efficient tool for the reduction of the size of the decision rule set without loosing the quality of classification of new objects.

Texture recognition

Tests have been performed on the decision table containing 320 objects, 24 conditional attributes and one decision attribute describing the texture type. The reduct set for the whole table has been calculated. It consists of 4774 reducts

The reduct set for a random sample of decision table containing 60% of objects contained 2248 reducts. The decision rules set has been computed for the sample applying methods presented in [11], [20]. The quality of classification was 84%. Next the dynamic reducts have been calculated. It happened that one of them was of much better quality than the others, i.e. the stability coefficient of this reduct was much higher than the stability coefficients for the others. It only consists of 7 attributes. The decision rule set, generated on the basis only of the attributes from that dynamic reduct, has the quality of classification a bit higher than the set of all decision rules. The exact value was in the range from 86% to 89% for different strategies of the final decision prediction. We have repeated the experiment several times and the results were almost the same.

The experiment suggests that our method can be treated as a method of extracting the relevant features from a given set of attributes (see e.g. [5], [6], [7]). The size of decision rule set generated on the basis of these features is much smaller than the decision rule set generated from all attributes and this reduction is done without loosing the quality of classification.

We have also made experiments taking the best 3 dynamic reducts with the stability coefficients higher than 64%. The quality of classification of the decision rule set generated on the basis of these dynamic reducts was 84%, so a bit less than for the best dynamic reduct. It happened because the new reducts are not so stable and have introduced some noise to classification. This experiment is showing that our method can be used also in a careful tuning process of noisy features' elimination.

Medical data

In the case of medical data the problem was to extract the most relevant features from the conditional attributes. The set of all reducts has been calculated. It has about 3000 reducts. The dynamic reducts have been extracted from this huge set. One of them has much better quality than the others, i.e. the stability coefficient of this dynamic reduct was much higher than of the others. The conditional attributes from it appeared to be, as we were informed after the experiment by Dr Tsumoto, the most important for the physicians. The reason that the number of reducts was so large was also explained by him: we have received data without the most important (from the point of view of diagnosis) conditional attributes and the question was to find from the noisy data the most relevant features among the remaining conditional attributes used by physicians as complementary in diagnosis.

5 Summary

We have presented some methods for computing dynamic reducts and decision rules from these reducts. The results of our experiments can be summarised as follows:

- Our methods can be treated as a tool for relevant feature extraction from the given set of features (e.g. conditional attributes extracted from the medical data appeared to be the most important for physicians).

- Performed experiments have proved that our method allows to choose a few dynamic reducts (with high stability coefficients) out of a huge number of reducts. The quality of unseen object classification based on these dynamic reducts is much better than on the whole set of attributes especially when data are noisy (e.g. market data). In other cases the quality of classification received by decision rules computed from the best dynamic reducts was of the same order as the classification quality of decision rules generated by all conditional attributes. We obtained a substantial reduction of the size of decision rules set without loosing the quality of classification (e.g. handwritten digits and texture recognition).

We continue the research trying to improve developed so far methods by new methods related to data filtration, reasoning with incomplete data (e.g. *null value*), and evolutionary learning based on rough sets.

This work was partially supported by the grant 8S503-019-06 from State Committee for Scientific Research (Komitet Badań Naukowych).

References

1. Bazan J., Skowron A., Synak P.: Market data analysis: A rough set approach, ICS Research Report **6**/94 Warsaw University of Technology 1994.
2. Brown, E.M.: Boolean reasoning. Dordrecht: Kluwer 1990.
3. Market data, Hughes Research Laboratories, manuscript.
4. International Workshop Rough Sets: State of the Art and Perspectives, Poznan - Kiekrz (Poland), September 2-4, 1992. Extended Abstracts. Poznan 1992. Full papers in Foundations of Computing and Decision Sciences, Vol.18, No **3-4** (1993).
5. Kodratoff Y., Michalski R.(eds.): Machine Learning vol III , Morgan Kaufmann, San Mateo, CA, 1990.
6. Michalski R., Carbonell J.G., Mitchel T.M. (eds.): Machine Learning vol I, Tioga/Morgan Kaufmann, Los Altos, CA, 1983.
7. Michalski R., Carbonell J.G., Mitchel T.M. (eds.): Machine Learning vol II, Morgan Kaufmann, Los Altos, CA, 1986.
8. Michalski R., Tecuci G.: Machine Learning. A Multistrategy Approach vol.IV, Morgan Kaufmann 1994.
9. Nguyen T., Swiniarski R., Skowron A., Bazan J., Thyagarajan K.: Application of rough sets, neural networks and maximum likelihood for texture classification based on singular value decomposition, submitted for the Workshop RSSC'94, San Jose, California 1994.
10. Proceedings of the International Workshop on Rough Sets and Knowledge Discovery, RSKD'93, Banff, October 11-15, Canada 1993, 101-104.
11. Pawlak Z. and Skowron A.: A rough set approach for decision rules generation. ICS Research Report **23**/93, Warsaw University of Technology 1993, Proc. of the IJCAI'93 Workshop W12: The Management of Uncertainty in AI, France 1993.
12. Pawlak Z.: Rough sets: Theoretical aspects of reasoning about data. Dordrecht: Kluwer 1991.

13. Quinlan J. R.: The Effect of Noise on Concept Learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchel (eds.), Machine Learning: An Artificial Intelligence Approach vol.II, Morgan Kaufmann Publishers, San Mateo, California 1986, 149-166.
14. Quinlan J. R.: Probabilistic Decision Trees. In: Kodratoff Y., Michalski R. (eds.) Machine Learning: An Artificial Intelligence Approach vol. III, Morgan Kaufmann Publishers, San Mateo, California 1990, 140-152.
15. Quinlan J. R.: C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, California 1993
16. Raedt De L.: Interactive Theory Revision. An Inductive Logic Programming, Academic Press 1992.
17. Shavlik J.W., Dietterich T.: Readings in Machine Learning, Morgan Kaufmann 1990.
18. Shrager, J., Langley, P.: Computational models of scientific discovery and theory formation, Morgan Kaufmann, San Mateo 1990.
19. Skowron, A. and Rauszer C.: The Discernibility Matrices and Functions in Information Systems. In: R. Slowinski (ed.): Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory. Dordrecht: Kluwer 1992, 331-362.
20. Skowron, A.: Boolean reasoning for decision rules generation. Proceedings of the 7-th International Symposium ISMIS'93, Trondheim, Norway 1993, In: J. Komorowski and Z. Ras (eds.): Lecture Notes in Artificial Intelligence, Vol.689, Springer-Verlag 1993, 295–305.
21. Slowinski, R. (ed.): Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory, Dordrecht: Kluwer 1992.
22. Swiniarski R., Nguyen T.: Application of rough sets, neural networks and maximum likelihood for texture classification based on singular value decomposition, Research Report SDSU, San Diego State University 1994.
23. Tsumoto S.: Medical Data, personal communication.
24. Trung, Nguyen, Tuan and Son, Nguyen, Hung: An Approach to the Handwriting Digit Recognition Problem Based on Modal Logic, ICS Research Report 44/93, Warsaw University of Technology 1993, M. Sc. Thesis, Warsaw University 1993.

Learning First Order Theories

Marco Botta
Dipartimento di Informatica - Università di Torino
Corso Svizzera 185 - 10149 Torino - ITALY
Tel.: (+39) (11) 7429221 - Fax: (+39) (11) 751603 - E-mail: botta@di.unito.it

Abstract. In the last decade, many efforts have been devoted to the exploration of techniques for learning and refining first order theories, as the necessary step for applying machine learning methodologies to real world applications. In this paper, we present a new approach to the integration of inductive and deductive learning techniques that seems to overcome some of the limitations of existing learning systems without imposing strong constraints or biases on both the representation language and the search space. In particular, a new search structure that enables the system to learn a structured knowledge base is proposed. Moreover, the learning system described in the paper can be used both to learn new knowledge from scratch and to refine an existing background theory.

1. Introduction

Studies in learning first order logic representations has followed two main trends: on one hand, many efforts have been devoted to build learning systems that are able to synthesize logic programs [5, 6, 16, 19]; on the other hand, the ability to learn and refine knowledge bases represented in first order logic languages is seen as the necessary step towards a broader class of applications.

In this paper we present a prototype system, KBI (Knowledge-Based Induction), that learns a knowledge base represented in a first order *function-free* logic language, takes full advantage of available a-priori knowledge and proposes a learning framework in which several learning tasks can be easily addressed, such as knowledge refinement and multiple concept learning. One goal of this work was to set up the learning framework in such a way that it can be easily extended in order to face deficiencies of existing systems (e.g., working with recursive background theories, dealing with noisy data, etc.).

What differentiates KBI from other relational learning systems (e.g., [3, 5, 6, 12, 16, 17, 18, 19, 23]) is not a brand new search algorithm, but, rather, a more informative search structure, called LT-Tree (Learned Theory Tree), that enables the system to learn a structured knowledge base limiting the complexity of the search process. The LT-Tree is not a new concept: Feldman et al. [7] addressed the problem of refining a domain theory, by proposing an incremental algorithm that modifies a special data structure, called *DT tree*: a DT_p *tree* is an AND/OR tree representing a full expansion of the domain theory rooted at predicate p. WHY [22] uses an AND/OR graph representation of the domain theory, as the basis for its *justification structure*. In his Multistrategy Task-Adaptive learning system [24], Tecuci uses an AND/OR tree structure to represent explanations and plausible justifications of examples. Informally, an LT-Tree is a generalization of Feldman's DT-Tree to manage first order theories and is directly derived from WHY justification structure. In particular, the LT_p Tree for a goal concept p is an AND/OR tree representing the learned definition for p.

Like most relational learning systems [3, 6, 17, 18, 19], KBI performs a general to specific search in the hypothesis space, looking for clauses that *cover* positive instances and no negative ones, until all positive instances are possibly covered. Unlike all of these systems that learn fully operational clauses, building up a flat set of rules, intermediate features defined by the domain theory can be present in the learned clauses and the system, while learning a concept description, is allowed to change their definitions, according to the current learning goal.

The learning problem can be formulated as follows: given a set E of examples and counterexamples of a target concept C, a background theory T, that approximately defines a set of intermediate features, and, possibly, the target concept description, the system has to find a new theory T' that completely and consistently (this condition is weakened in presence of noisy data) defines the target concept C in terms of both operational and intermediate features.

The paper is organized as follows: Section 2 introduces the learning framework; Section 3 presents in more details the structure of the search tree, whereas in Section 4 the learning operators are discussed. Section 5 sketches the general learning algorithms. Experiments are reported in Section 6, whereas related works are discussed in Section 7, and, finally, Section 8 concludes pointing out current limitations and future developments.

2. The Learning Framework

A *normal program clause* is a logical expression of the form $p \leftarrow L_1 \wedge ... \wedge L_n$, where p is a positive atom and the L_i's are positive or negative atoms, called *literals*. $L_1 \wedge ... \wedge L_n$ constitutes the *body* of the clause and p is said the *head* of the clause. In the following we will use the terms atom and predicate interchangeably. Predicates can be defined *extensionally*, by specifying a list of tuples for which the predicate is true as a database relation, or *intensionally*, as a set of normal program clauses for computing whether the predicate is true. Extensionally defined predicates are also said to be *operational*, that is directly evaluable on the data, whereas intensionally defined predicates are said to be *non-operational*, that is they need a more complex deductive procedure in order to be proved. Predicates with variables are permitted in a clause, but they may not contain neither function symbols nor constants. Furthermore, normal program clauses considered in this work are *range-restricted*, i.e., all variables occurring in the head of the clause also occur in its body. A set of normal program clauses that share the same head p is a *rule* for p. A *theory*, as intended in this work, is a collection of rules for a set of non-operational predicates $\mathbf{P}^{(n)}$. A theory, in order to be tractable by the system, must be stratified [1]. Stratification determines priority levels (strata) of a set of predicates **P**, with the lowest level containing operational predicates. This condition guarantees that a predicate is not recursively dependent on its negation. In the following, we will adopt the closed world assumption [20] and the negation as failure rule [4], in order to associate a semantics to the predicates.

The system architecture is reported in Fig. 1: two main modules interacts with each other, receiving inputs from a domain theory and a relational database and producing a target knowledge base.

The relational database hosts three kinds of relations: an OBJ relation describes the learning instances; a CLASS relation represents the target concept relation; FORMULA relations contain the extension of logical formulas on the data. The extensional theorem prover consists of a deduction procedure that receives queries from the Learning Module and instructs a database manager to compute the extensional representation of hypotheses.

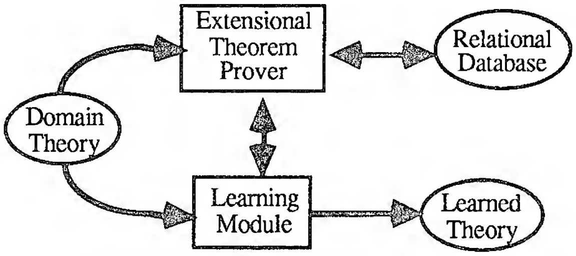


Fig. 1 - System architecture.

The deduction procedure implements the resolution principle in a bottom-up fashion similar to the OPS5 inference engine [9]. Two characteristics of bottom-up theorem provers motivated this choice: bottom-up methods are generally more efficient than top-down ones; moreover, they are complete for stratified programs. This second property allows recursive predicates in the domain theory to be managed. In particular, the theorem prover builds up extensions of non-operational predicates in a context-free manner and stores the resulting relation in the database for future usage. Doing in this way, the deduction process can be performed only once for each non-operational predicate, the first time it is requested, with a great saving in computational costs, at the expense of a small increase in memory requirements. Even if works on speedup learning [8] showed that this strategy does not always lead to an improvement in efficiency and found conditions under which this is true, that does not apply in this context. In fact, let us take a step ahead and consider how KBI learns a clause body: at first, all operational and non-operational predicates are tried and the one which shows the "best" gain is added to the current hypothesis. Hence, the extension of non-operational predicates is computed at least once; if the result is stored in the database, all subsequent uses do not have to repeat the deduction process, with a clear efficiency improvement. Therefore, as done by WHY [22], a pre-processing step is performed in order to compute the extension of all the non-operational predicates defined in the domain theory.

3. Description of the LT-Tree

The search tree used by KBI is an AND/OR proof tree of the goal concept that is to be learned. OR nodes correspond to literals, whereas AND nodes correspond to clauses. The root of the tree is an OR node corresponding to the concept to be learned. All leaves of the tree are OR nodes that correspond to operational predicates. The OR node for a predicate p has as many son AND nodes as the number of clauses in the theory T defining p, that have a non-null extension on the training instances. The AND node for a clause c has as many son OR nodes as the number of literals in its body. Different instantiations of the same literals are kept separate by creating a new subtree in the LT-Tree, with the exception of operational predicates whose OR nodes are not duplicated.

Given a goal concept G, the LT_G Tree is initialized in two steps: the first step builds up the initial tree structure, by mirroring the proof tree for G, and computes a preliminary extension of the nodes in a bottom-up fashion, starting from the leaves. At the end, every node in the LT_G Tree has an associated extension in the database.

But only the bindings for variables in the goal concept (the root node) determine the learning problem. Therefore, in the second step, the system propagates downward these bindings, by eliminating from every extension those tuples that are inconsistent with these bindings, with two main advantages: first of all, the cardinality of relations down the tree is reduced; secondly, the system can focus its attention on the relevant bindings at any level in the search tree. This also allows to easily estimate the effect of a modification to the tree on the extension of the goal concept.

There are a number of advantages in using this structure with respect to traditional ones: each clause contains fewer variables than any "big" fully operational hypothesis, with a reduction in the size of database relations containing the extension of hypotheses on the learning instances and in the number of variabilization of literals to test during learning; any clause can be locally changed without affecting its definition both in the background theory and in other branches of the LT-Tree (only after the learning phase, the new theory T' is substituted for the old one); this structure is open to the use of powerful heuristics (more informative than information gain based ones) to guide the search process. On the contrary, drawbacks are the greater complexity of the search process, because there are many more points to take under consideration, and the need of more complex updating procedures (anyway linear in the number of nodes), to keep the structure up to date. Besides, without posing restrictions, the number of nodes in an LT-Tree can become very large even when there are few non-operational predicates defined in the theory T; however, as it has been observed experimentally in applications in which $|P^{(n)}| \approx 15$, $|P^{(o)}| \approx 25$ and $|T| \approx 50$, an LT-Tree never contains more than 100 nodes.

4. Learning Operators

At any time, an LT-Tree represents a partial definition of the goal concept that may be inconsistent with the given instances. The search tree can be modified by performing one of the following operations:

- Specialization by adding a literal to a clause (AND node) c involves two steps: selecting a literal to be added and updating the search structure. Many existing systems can only select literals from the set of operational predicates (some are further restricted to positive literals), whereas KBI, like FOCL [17, 18] and WHY [22], can select both positive and negative literals from the operational and non-operational sets of predicates. A second difference between KBI and other systems concerns variabilization of literals to be added: in principle, given a clause c KBI does not require that a literal to add to the body of c be *determinate*, and therefore new variables that are completely unrelated to the existing ones can be introduced. This is because in some situations (see [2], for an example) determinate literals prevent finding a solution.
- Specialization by deleting a clause (AND node), i.e., by eliminating one of the alternative ways to prove a predicate p in the search tree. In this way, it is possible to recover from incorrectly overly general rules defined in the background theory. A clause may be deleted if there is at least one sibling AND node in the search tree that defines the same predicate.
- Simplification by collapsing two predicates can be applied to the tree when an AND node satisfies the following three conditions: a) there are no sibling AND nodes; b) the associated extension contains only positive instances and c) the corresponding clause c has one of the following forms:

$$P(x_1, ..., x_n) \leftarrow Q(x_1, ..., x_n)$$

$$P(x_1, ..., x_n) \leftarrow \neg Q(x_1, ..., x_n)$$

i.e., P is defined either in terms of Q or of the negation of Q. Then, the OR node corresponding to P and the AND node corresponding to c can be eliminated and the occurrence of $P(x_1, ..., x_n)$ in the upper clause is substituted by the body of c. The simplification operation has been introduced in order to deal with overly general clauses and can be performed as a post-processing step on the resulting tree.

- Generalization by deleting a literal from a clause (AND node) allows to deal with overly specific theories, where clauses may contain superfluous or incorrect literals. Given an AND node in the tree, a local search is performed on the corresponding clause, in order to find if one or more of the literals in the body of the clause can be deleted: in particular, a literal can be deleted only if positive tuples are possibly added to the extension of the node (i.e., if no new specialization problems are introduced).

5. Global Learning Strategy

The main learning cycle is sketched in Fig. 2, and consists of two nested loops: the outer loop is responsible to restart the search for a new set of clauses until all examples are covered; the inner loop builds up an LT-Tree and performs a search process aimed to find a consistent rule for the goal concept.

```
Algorithm Main-Loop (Goal-Concept)
    concept-descrs = { }
    while there are positive examples not yet covered do
        tree = build up an initial LT$_{Goal-Concept}$ Tree
        while tree covers negative examples do
            todo = Select-Operation (tree)
            tree = Apply-Selection (tree, todo)
            tree = Simplify-Tree(tree)
        enddo
        tree = Try-Generalize (tree, concept-descrs)
        declare covered positive examples in tree
        concept-descrs = concept-descrs $\cup$ {tree}
    enddo
return concept-descrs
```

Fig. 2 - Main search algorithm.

When a complete and consistent concept description cannot be found the termination conditions in the while statements can be weakened by introducing threshold values. At the end of the inner cycle of Main-Loop, `tree` contains a definition of the goal concept that should cover at least one instance that was not covered by previously learned concept descriptions. If this is not the case, it is not possible to find a complete concept description in the given language. One peculiarity of this algorithm is the generalization step performed by Try-Generalize after exited the inner loop: it may be the case that the hill climbing strategy used to modify the search tree led to an overspecialization of the target concept description. Try-Generalize looks for literals in the tree that if deleted result in covering more examples without covering more counterexamples. Try-Generalize may also be more sophisticated and try to perform a global generalization on the partial concept description represented by `concept-descrs` and `tree`, by *blending* the trees generated so far (not implemented, yet).

One difference between KBI and FOIL-like systems is the use of instances: when a rule is found and the inner loop is exited, positive examples are not removed from the learning set, as done in FOIL, but are declared *covered*, with two advantages: more complex information measures can be used (e.g., the total number of instances in the learning set that verify a hypothesis can be taken into account, along with the number of instances not yet covered) and incorrect hypotheses generation is avoided. By only marking instances as covered, the focus of attention of the system can still be biased towards not yet covered instances, without losing the possibility to check for consistency with the whole learning set. If a conjunctive concept description exists, then it is found in the inner loop of the Main-Loop algorithm and the search is terminated. Otherwise, each conjunct is learned by repeating the inner loop as many times as needed, each time focusing the search on instances not yet covered.

The most crucial part of algorithm Main-Loop is the selection of how to modify the search tree. The *Select-Operation* function performs two steps, as described in Fig. 3: in the first step, the search tree is traversed and every AND node is evaluated in order to decide whether to further specialize (and applicable predicates are collected) or delete the node; in the second step, all trials are scored according to a criterion that combines an information gain measure with an evaluation of the generality of the operation, and the best scored trial is returned. For example, when KBI has to choose among operational and non-operational literals that show the same information gain measure, non-operational literals are considered more general and, therefore, preferred for specialization.

```
Function Select-Operation (Tree)
Step 1: toeval = {}  Toeval is a list of pairs (c, L) where c is a clause in Tree and L
                     is either a literal or the keyword delete
   for every clause c in Tree do
      if c covers many counterexamples and few examples then
         toeval = toeval ∪ (c, delete)
      else if c covers counterexamples then
         toeval = toeval ∪ Try-Specialize (c)
   endfor
Step 2: best = (c, L) | ∀ (c',L') ∈ toeval : score((c, L)) > score((c',L'))
return best
```

Fig. 3 - Algorithm that selects what to do at each iteration of the Main-Loop. Keywords *many* and *few* are thresholds that account for minimal completeness of hypotheses.

Notice that, in step 1, clauses that cover few instances (either examples or counterexamples) are not considered for any specialization operation. These clauses will be analyzed by the generalization step performed by Try-Generalize, after the inner loop is exited.

Function *Try-Specialize* takes as input a clause and returns all the literals (operational and non-operational) that added to the clause results in a non-empty extension. Try-Specialize works as a two step process: in the first step, all variabilizations for a predicate p are constructed, whereas, in the second step, an estimation of the resulting extensions is computed. There are a number of optimizations that can be applied to this step in order to limit the amount of literals to be evaluated and to improve efficiency. First of all, variabilizations of a predicate that result in a null extension for a given clause c, need not be proved again in later repetitions of the inner loop. Secondly, variabilizations of predicates can be computed the first time the clause is considered for specialization, and updated only when new variables are introduced in the body of the clause. Finally, *Apply-Selection* modifies the LT-Tree by either adding or deleting a subtree corresponding to the selected literal or clause, respectively.

6. Experiments

That the integration of inductive and deductive learning techniques produce better results than any of the two separately, has already been assessed by a number of authors (see, for instance, [17, 21, 15]). The following experiments reinforce this view, while showing that KBI control strategies allow the system to reach good performances even when few instances are provided and that the LT-Tree search structure does not result in a significant computational complexity growth (a theoretical analysis is undergoing).

6.1 Chess Endgames

The first set of experiments concerns the popular chess endgame domain White King and Rook versus Black King: given board positions for white king, white rook and black king learn conditions that state if an unseen board is not a legal white-to-move board. There are a number of situations that generate an illegal board position: for each of these situations a non-operational predicate has been defined in the domain theory provided to the system. In particular, the following intermediate features have been defined: King_attack_King (kak), King_not_between_{rank,file} (knotbet), Rook_attack_king (rak), same location (sameloc), between_{rank,file} (Between). Moreover, 10 operational predicates were defined ({eq,adj,less}_{rank,file}, {white, black}_king, white_rook and diff}.

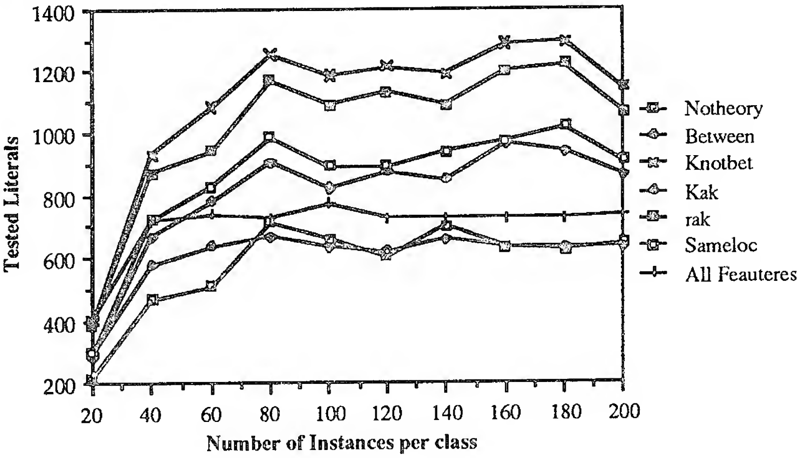


Fig. 4 - Number of tested literals in learning Illegal.

Fig. 4 reports the number of tested literals (averaged over five runs) when using different domain theories, defining one of the previous mentioned predicates, and increasingly larger, randomly generated training sets are provided. As can be noted, the number of tested literals depends on the number of training instances only when few instances are supplied; instead, the impact of the theory on the number of tested literals is not uniform and strongly depends on the defined features: only in two cases (kak and All Features) the number of tested literals is close to the one with no theory at all; anyway, all values are comparable with FOIL's and FOCL's, as reported in [17].

For what concerns the classification rate, the learned knowledge bases have been tested on a set of 1000 independently, randomly generated instances (500 Illegal and 500 Legal board positions). Rows in Table I refer to the size of training sets, whereas columns report the correct classification rate of the knowledge learned from six different domain theories. As one can expect, the influence of the theory is greater when few training instances are provided. It should be noted that with few randomly generated instances, it is unlikely to obtain 100% correct classifications, even when all intermediate features are defined, because some situations may not be represented in the training instances.

Table I - Correct classifications for Illegal.

Instances per class	Notheory	kak	knotbet	rak	sameloc	All Features
20	78.9	83.7	78.7	84.2	78.7	93.1
40	93.8	99.4	93.8	93.8	93.8	100.0
80	99.4	99.4	99.4	100.0	99.4	100.0
160	99.6	99.6	99.6	100.0	99.6	100.0
200	99.6	99.6	99.6	100.0	99.6	100.0

6.2 Thousand Trains Problem

The thousand trains problem [11] is an extension of the well known train going east and west test set defined by Michalski [13]. The learning set used in this experiment consists of 1000 instances, 500 of which are classified as trains going east, according to three manually generated rules. A domain theory defining a set of 12 intermediate features, such as a {short, long} coach with an {open, closed} top, a loaded coach, etc., is also provided to the system. As the learning task can be solved without the use of the domain theory, the aim of this experiment was to empirically evaluate the overhead introduced by the non-operational features. Since the total number of possible combinations of non-operational predicates is 2^{12}, we selected ~10% of combinations and averaged the results, reported in Table II.

Table II - Number of tested literals using 0-6 and 12 features.

0	1	2	3	4	6	12
7423	5842	3940	3929	3490	1810	2058

The use of the domain theory makes the search process easier in two respects: without theory at all, the system found a large number of small disjunct, whereas with all the features defined, KBI discovered just top level rules, two of which are slightly more general than those used to label the instances; moreover, the number of tested literals decreases as the number of features increases, and is reduced to 1/3 when all features are defined.

7. Related Works

FOIL [19] learns classification rules by constructing a set of Horn Clauses in terms of known operational predicates. Each clause body consists of a conjunction of literals that covers some positive and no negative examples. The learning process ends when all positive examples have been covered by some clause. Should FOIL be provided with the extension of non-operational predicates defined in a domain theory, the main difference between FOIL and KBI would be the control strategy used to perform the search that, in KBI, is less greedy and allows to dynamically change the definition of such predicates.

FOCL [17] extends FOIL by incorporating a compatible EBL component. This allows FOCL to take advantage of an initial domain theory. When constructing a clause body, there are two ways that FOCL can add literals. First, it can create literals via the same empirical method used by FOIL. Second, it can create literals by operationalizing a target concept, i.e., a non-operational definition of the concept to be learned [14]. However, clauses learned by FOCL are fully opeartional, i.e. contain only operational predicates, whereas KBI learns a structured knowledge base. Moreover, unlike EBL-derived (or based) integrated learning systems, KBI does not require an approximate target concept description in order to use the theory.

FOCL-Frontier [18] further extends FOCL by using an analytic learner that does not necessarily fully operationalize target concepts. Instead, the learner returns a *frontier* of the domain theory. A frontier differs from an operationalization of a domain theory in three ways: a) non-operational predicates can appear in the frontier; b) a disjunction of two or more clauses that define a non-operational predicate can appear in the frontier; c) a frontier does not necessarily include all literals in a conjunction. It seems that the computation of a frontier of the domain theory is very much like learning a LT_p Subtree in KBI, for a non-operational predicate p in the context of the current hypothesis.

KBI is, in some respect, similar to theory revisions systems, like EITHER [15], FORTE [21], RTLS [10], AUDREY II [25] and others, but it does not have the goal of making minimal revisions to a theory; even if our claim is that the theory revision task can be addressed in the presented framework, more work needs to be done on defining KBI limits at this task, specifying how and when the learned theory should replace the old one and there is the need to refine the control strategies.

8. Conclusions

A new learning framework, based on the LT-Tree concept, has been presented, along with a general effective learning algorithm, that takes full advantage of the proposed search structure. KBI has been successfully experimented on simple learning domains, obtaining encouraging results. The current implementation has a number of limitations but shows many potentialities that will be explored in the future. In particular, since KBI is able to manage recursive predicates in the domain theory, it would be desirable to learn recursive definitions. Moreover, further efforts will be devoted to incorporate the ability to deal with numerical features, in the line of [3], to learn multiple concepts and subconcepts, and to perform extensive tests on more complex domains.

References

1. Apt, K.R., Blair, H.A., and Walker, A. "Towards a Theory of Declarative Knowledge", in *Foundations of Deductive Databases and Logic Programming*, Minker, J. (Ed), Morgan Kaufmann, Los Altos, CA, 89-148, 1988.
2. Baroglio, C., Botta, M., and Giordana, A. "Learning Relations: An Evaluation of Search Strategies", *Fundamenta Informaticae*, *18*, No. 2,3,4, 221-232, Feb.-Apr., 1993.

3. Botta, M. and Giordana, A. "Smart+: A MultiStrategy Learning Tool", *Proc. of the IJCAI-93*, Chambery, France, August, 937-943, 1993.
4. Clark, K. "Negation as Failure", in *Logic and Data Bases*, Gallaire, H. and Minker, J. (Eds), Plenum Press, New York, 275-295, 1978.
5. De Raedt, L. and Bruynooghe, M. "Towards Friendly Concept-Learners", *Proc. of the IJCAI-91*, Sidney, Australia, 849-854, 1991.
6. De Raedt, L., Lavrač, N., and Džeroski, S. "Multiple Predicate Learning", *Proc. of the IJCAI-93*, Chambery, France, 1037-1042, 1993.
7. Feldman, R., Segre, A., and Koppel, M. "Incremental Refinement of Approximate Domain Theories", *Proc. of the Eigth International Workshop on Machine Learning*, Evanston, IL, 500-504, 1991.
8. Fisher, D., Subramanian, D., and Tadepalli, P. (Eds). *Proceedings of the Third International Workshop on Knowledge Compilation and Speedup Learning*. Amherst, MA, 1993.
9. Forgy, C.L. "RETE: A Fast Algorithm for the many Pattern/many Object Pattern Match Problem", *Artificial Intelligence*, *19*, 17-37, 1982.
10. Ginsberg, A. "Theory Revision via Prior Operationalization", *Proc. of the AAAI-88*, Saint Paul, MN, 590-595, 1988.
11. Giordana, A. and Saitta, L. "REGAL: An Integrated System for Learning Relations using Genetic Algorithms", *Proc. of the Second International Workshop on MultiStrategy Learning*, Harpers Ferry, WV, 234-249, 1993.
12. Lavrač, N., Džeroski, S., and Grobelnik, M. "Learning non-recursive definitions of relations with LINUS", *Proc. of the EWSL-91, LNAI 482*, Springer-Verlag, Porto, Portugal, 1991.
13. Michalski, R.S. "Pattern recognition as a rule-guided inductive inference", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-2*, 349-361, 1980.
14. Mitchell, T., Keller, R., and Kedar-Cabelli, S. "Explanation Based Generalization: A Unifying View", *Machine Learning*, *1*, 47-80, 1986.
15. Mooney, R.J. and Ourston, D. "A Multistrategy Approach to Theory Refinement", in *Machine Learning: A Multistrategy Approach*, Vol. IV, Michalski, R.S. and Tecuci, G. (Eds), Morgan Kaufmann, San Francisco, CA, 141-164, 1994.
16. Muggleton, S. and Feng, C. "Efficient Induction of Logic Programs", *Proc. of the First Conference on Algorithmic Learning Theory*, Tokyo, Japan, 1990.
17. Pazzani, M.J. and Kibler, D. "The Utility of Knowledge in Inductive Learning", *Machine Learning*, *9*, 57-94, 1992.
18. Pazzani, M.J. and Brunk, C. "Finding Accurate Frontiers: A Knowledge-Intensive Approach to Relational Learning", *Proc. of the AAAI-93*, Washington, D.C., 328-334, 1993.
19. Quinlan, R. "Learning Logical Definitions from Relations", *Machine Learning*, *5*, 239-266, 1990.
20. Reiter, R. "On Closed World Data Bases", in *Logic and Data Bases*, Gallaire, H. and Minker, J. (Eds), Plenum Press, New York, NY, 55-76, 1978.
21. Richards, B. and Mooney, R. "First-Order Theory Revision", *Proc. of the Eight International Workshop on Machine Learning*, Evanston, IL, 447-451, 1991.
22. Saitta, L., Botta, M., and Neri, F. "Multi-strategy learning and theory revision", *Machine Learning*, *11*, 153-172, 1993.
23. Shapiro, E.Y. *Algorithmic Program Debugging*, The MIT Press, 1983.
24. Tecuci, G. and Duff, D. "Knowledge Base Refinement Through a Supervised Validation of Plausible Reasoning", *Proc. of the Second International Workshop on MultiStrategy Learning*, Harpers Ferry, WV, 76-91, 1993.
25. Wogulis, J. and Pazzani, M. "A Methodology for Evaluating Theory Revision Systems: Results with Audrey II", *Proc. of the IJCAI-93*, Chambery, France, 1128-1134, 1993.

Evaluation and Enhancement of Bayesian Rule-Sets in a Genetic Algorithm Learning Environment for Classification Tasks

Christoph F. Eick and Ema Toto

Department of Computer Science
University of Houston, Houston TX 77204-3475
e-mail: ceick@cs.uh.edu
phone: (713) 743-3345; fax: (713) 743-3345

Abstract. The paper describes a learning environment named DELVAUX for classification tasks that learns Bayesian rule-sets from sets of examples. A genetic algorithm approach is used for this purpose, in which a population consists of sets of rule-sets that generate offspring through the exchange of rules, permitting fitter rule-sets to produce offspring with a higher probability. A bucket brigade algorithm for Bayesian rule-sets called reward punishment mechanism is introduced, which evaluates the performance of a Bayesian rule within a rule-set. It employs fuzzy techniques to measure the "goodness" of a rule within a rule-set.

1 Introduction

Bayesian rules rely on interpolation, approximate reasoning techniques, and decision making by evidence combination: each rule provides evidence for or against a particular classification; the evidence obtained from different rules is then combined, and the classification with the highest probability is selected. This paper centers on the development of a machine learning environment that learns Bayesian rule-sets for classification tasks using a genetic algorithm approach. Rule learning has already intensively been studied in genetic algorithm research in a subfield called *classifier systems* [DJ90, JA93, OL93]. In these systems populations consist of a single rule-set, in which rules are evaluated with the help of the well known bucket brigade algorithm [HO86], and the new generation is generated by applying genetic operators, such as crossover and mutation, to the current population. However, the rules learnt by classical classifier systems are purely symbolic and rely on two-valued classical logic — their left-hand side is either true or false.

One critical problem for approaches that try to learn rule-sets from sets of examples is the complexity of the search space. Evaluation and fitness-based exploration of the search space play a key role for increasing the likelihood that more interesting solutions are found during the search process. However, not only the evaluation of a solution, but also the evaluation of sub-components of a solution might be helpful to improve the search process. The latter evaluation can be used to guide the search process so that bad sub-components of a solutions are modified with higher probability than good sub-components, making the search process more directed.

This paper adopts the latter idea of evaluating sub-components of solutions to improve the search process for a good Bayesian rule-set. More specifically, a bucket-brigade algorithm for Bayesian rules, called *reward punishment mechanism* (RPM), is introduced in this paper that evaluates individual rules within a rule-set based on their classification performance. Moreover, the features of a learning environment called DELVAUX which employs the reward punishment mechanism are briefly introduced.

2 Bayesian Classification Rules

This section intuitively introduces the semantics of Bayesian classification rules that we try to learn (a more detailed discussion of these topics and of the DELVAUX environment has been given in [EJ93]).

Throughout this paper we will use an Iris-flower classification problem to illustrate the features of DELVAUX: based on sepal length(SL), sepal width(SW), pedal length(PL), and pedal width(PW) it has to be decided, if a given Iris-flower is a Setosa, Versicolor, or Virginica. Moreover, we assume that the values of the various attributes have been normalized to be in $[0,1]$, measuring how high attribute value is in comparison to the values of the same attribute for the other example[1]; consequently, 0 is considered to be very low and 1 is considered to be very high value of an attribute.

A set of training examples is given to our learning environment; in the case of the Iris-flower classification problem, each example consists of the values of the four attributes and the class the particular example belongs to as depicted in Fig. 1 (e.g. the first example is a Setosa which has the value 0.3 for attribute SL, 0 for PL, 1.0 for SW, and 0.67 for PW).

In general, rules used in our learning environment work with odds instead of probabilities, using the following conversion from probabilities to odds

$$O(H) := \frac{P(H)}{1 - P(H)}$$

Initially, only the prior odds of being a particular flower are given — in our example we assume these are:

[1] In general the following conversion function Γ is used to convert the values of an attribute A:

$$\Gamma_A(a) = \frac{Number\ of\ Examples\ whose\ value\ for\ A\ is\ less\ than\ a}{Number\ of\ Examples\ whose\ value\ for\ A\ is\ different\ from\ a}$$

For example, assuming that we have a set of 10 examples which have the following values for an attribute A

1,1,2,3,3,3,3,4,4,4

we receive:

$\Gamma_A(1) = 0, \Gamma_A(2) = \frac{2}{9}, \Gamma_A(3) = \frac{3}{6} = 0.5,\ and\ \Gamma_A(4) = 1.0$

```
                (SL,  PL,  SW,  PW,  CLASS)
Example 1 : (0.3, 0.0, 1.0, 0.67, Setosa)
Example 2 : (0.3, 0.7, 0.3, 1.0, Versicolor)
Example 3 : (0.9, 0.0, 0.0, 0.37, Virginica)
```

(a) Training Data Set

```
(R1) If is-high(SW) then Setosa with S=5, N=0.2
(R2) If is-high(PL) then Versicolor with S=12, N=0.1
(R3) If is-close-to(SL,0.3) then Setosa with S=5.5, N=0.8
(R4) If is-high(PW) then Virginica with S=0.2, N=4
(R5) If is-high(SL) then Virginica with S=4, N=0.1
```

(b) Bayesian Rule-Set

Figure 1 : Example of Training Set and Bayesian Rule-Set

$O(Setosa) = 1, O(Virginica) = 0.67, O(Versicolar) = 0.11$

that is, 50% of Iris-flowers are Setosa, 40% Virginica, and 10% are Versicolors. These odds are then updated by the Bayesian rules in the light of new evidence (in our case new evidence means interpreting the sepal length, sepal width, pedal length or pedal width of a particular example), yielding the a posteriori odds for each class of Iris-flower. In the above, S and N are odds-multipliers, measuring the *sufficiency* and *necessity* of right-hand side conclusion for the left-hand side condition. The multiplier S is used when the the left-hand condition of the rule evaluates to true, N is used if it evaluates to false.

Let us explain, how the first example $(0.3, 0.0, 1.0, 0.67, Setosa)$ as classified by the above rule-set. The first rule states that if the third field SW is high then the odds of Setosa should be increased by the multiplication of 5, if SW is not high the odds of Setosa should be decreased by the multiplication of 0.2. Because SW is 1, the first rule computes a multiplier of 5. The second rule analyzes the second field PL, and its value is 0.0; consequently, the rule's N value is used, and the odds of Versicolor are decreased by of a factor of 0.1.

In general, our learning environment supports two kinds of rules: close-to and is-high rules. The third rule is a close-to rule; its left-hand side condition checks if a particular attribute value is close to a constant. In our example rule-set the third rule checks if the first field sepal length is close to the constant 0.3; because this is the case a multiplier of 5.5 is derived for the example.

The fourth rule checks if the fourth field PW is high; however, the value of the fourth field is 0.67; that is, it is neither 0 or 1. In this case, interpolation has to be used to compute the rule's multiplier — various interpolation techniques have been proposed in the literature for this purpose. For example, the following simple linear interpolation function might be used

$0.67 * 0.2 + 0.33 * 4 \approx 1.45$

yielding a multiplier of 1.45 for the fourth rule. Finally, a multiplier of 1.27 is computed for the fifth rule. After all rules have been fired, the a posteriori odds of the decision candidates are computed using:

$$O(D') = O(D) * \prod_{r \in R_D} \lambda_r$$

that is, the posteriori odds of the decision candidate D (O(D')) are computed by multiplying the prior odds of D (O(D)) with with the multipliers of all rules that provide evidence for D (this rule-set is denoted by R_D in the above formula). We obtain the following a posteriori odds for the three decision candidates for the first example:

$O(Setosa') = 1 * 5 * 5.5 = 27.5$
$O(Versicolor') = 0.11 * 0.1 = 0.011$
$O(Virginica') = 0.67 * 1.45 * 1.27 = 1.23$

Because Setosa has the highest a posteriori odds, the example is classified as a Setosa, which is the correct answer. In summary, the task to be accomplished by our DELVAUX learning environment to be described in the next section is to learn a "good" Bayesian rule-set consisting of is-high and close-to rules from a set of training examples.

3 Fuzzy Evaluation Techniques for Individual Rules

Before we will describe our techniques to evaluate the performance of individual rules within a Bayesian rule-set, we will introduce the basic features of the DELVAUX learning environment in which these evaluation mechanisms are employed.

3.1 Architecture and Features of DELVAUX

Figure 2 gives the basic architecture of the DELVAUX environment: Using a set of training examples as an input a Bayesian rule-set is learnt, this rule-set is then evaluated for a set of test examples.

In our approach, populations consists of sets of Bayesian rule-sets. Fitness of a rule-set is evaluated in our approach by the percentage of examples the rule-set classifies correctly. This fitness function is denoted by h in the following. For example, if our training set consists of 80 examples, and a rule-set RS classifies 70 of these examples correctly, RS's fitness would be: $h(RS) = 70/80 = 0.875$.

Our initial population — the first generation — is generated randomly. The next generation is generated by using 1-point crossover operators, taking some rules from one parent and some rules of the other parent. Parents are selected based on their fitness values (which are computed by applying a fitness function h to the rule-set) using the popular roulette wheel method. Occasionally, rule-sets are mutated, by replacing a randomly chosen rule by a new, randomly generated rule.

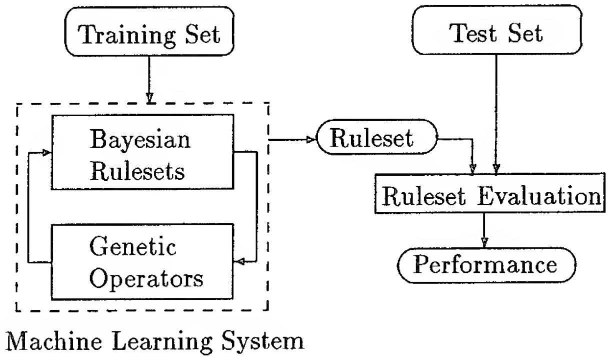


Figure 2 : Architecture of DELVAUX

3.2 Rewarding Individual Rules

The basic DELVAUX learning environment evaluates rule-sets by using the fitness function h. In this section, we introduce a new credit assignment mechanism, at the rule's level, called the *reward punishment mechanism* (RPM). It evaluates individual rules within rule-set, assigning a score to each of them depending on their performance in the rule-set's decision making.

In our Bayesian model, a rule provides evidence for its decision, with respect to a given example case. Intuitively, we can talk about bad rules and good rules: a good rule provides positive evidence for the class particular example belongs to, or it provides negative evidence for a class the example does not belong to, whereas bad rules show the opposite behavior.

More formally, let

R-Set $= \{r_1, r_2, \ldots, r_n\}$ be a rule-set with n rules,

TR-set $= \{tr_1, tr_2, \ldots, tr_m\}$ the training set with m examples, and

$\lambda_{i,j}$ be the multiplier of rule r_i for testcase tr_j.

Mathematically, the "goodness" of a rule can be considered to be a fuzzy set:

$$\tau_{good}(r) = \frac{number\ of\ times\ rule\ r\ worked\ correctly}{number\ of\ times\ rule\ r\ is\ fired}$$

In our approach, we approximate the above membership function τ with the help of *pay-off functions.* Pay-off functions distribute the feedback of the environment to individual rules of the classifier system. If the rule-set classifies an example correctly, every rule that contributed to making the correct decision will be rewarded from the environment. On the other hand, if the rule-set classifies an example incorrectly, rules that provide positive evidence for the incorrect chosen decision, and rules that provide negative evidence for the correct decision have to be punished. The rewards and punishments that a rule receives for the examples of the training set, will be added together, yielding the rule's *score.* Scores are relative measures: they describe how well a particular rule performs with respect to the other rules of the rule-set: rules with high scores are considered to be "good", whereas rules with low scores are considered to be "bad".

This scheme is implemented as follows:
Let $R = (r_1, r_2, \ldots, r_m)$ be a rule-set, $\{D_1, D_2, \ldots, D_l\}$ the classes and *Tr-Set* $= (tr_1, tr_2, \ldots, tr_n)$ the training set, with respect to which the rule-set R has to be evaluated. Our evaluation scheme uses two functions:

f' denotes a payoff function — it computes a penalty or a reward for a particular rule for a particular example. Each rule receives a separate payoff for each example, reflecting the rule's performance in classifying this example.

f computes a rule's score, which is the sum of a rule's payoffs over the examples of the training set. The a score of a rule measures the "goodness" of a rule. Rules within a rule-set that have high scores are considered to be better than rules that have low scores. That is:

$$f(r) = \sum_{i=1}^{n} f'(tr_i, r).$$

The definition of the payoff function f' refers to the following 3 rule-sets: $P_{k,i}$, $N_{k,i}$, and $ON_{k,i}$ (k denotes the index of a class, and i is the index of a training example). These three sets will be used by the payoff function f' to identify rules that have to be rewarded or punished in a particular context:

Let

$P_{k,i}$ be the set of rules $\{rp_1, \ldots, rp_{p_{k,i}}\}$ that provide positive evidence for class D_k of the i-th example of the training set, having multipliers

$$\lambda_{rp_1,i}, \ldots, \lambda_{rp_{p_{k,i}},i}$$

and $p_{k,i}$ be its cardinality. Moreover,

$$Mult(P_{k,i}) = \sum_{j=1}^{p_{k,i}} \lambda_{rp_j,i}.$$

$N_{k,i}$ be the set of rules $\{rn_1, \ldots, rn_{n_{k,i}}\}$ that provide negative evidence for class D_k of the i-th example of the training set, having multipliers

$$\lambda_{rn_1,i}, \ldots, \lambda_{rn_{n_{k,i}},i}$$

and $n_{k,i}$ be its cardinality. Moreover, let

$$Mult(N_{k,i}) = \sum_{j=1}^{n_{k,i}} \frac{1}{\lambda_{rn_j,i}}.$$

$ON_{k,i}$ be the set of rules $\{ron_1, \ldots, ron_{on_{k,i}}\}$ that provide negative evidence for all $D \neq D_k$ of the i-th example of the training set, having multipliers

$$\lambda_{ron_1,i}, \ldots, \lambda_{ron_{on_{k,i}},i}$$

and $on_{k,i}$ be its cardinality. Moreover, let

$$Mult(ON_{k,i}) = \sum_{j=1}^{on_{k,i}} \frac{1}{\lambda_{ron_j,i}}.$$

For example, $P_{1,5}$ contains all rules that provide positive evidence (have multipliers greater than 1) for class D_1 for the fifth example (tr_5), or $ON_{2,7}$ contains all rules that provide negative evidence (have multipliers less than 1) for example tr_7 for decisions $\{D_1, D_3, ..., D_l\}$ (the complete set excluding D_2).

Using the above definitions, we define the function $f'(tr_i, r)$ distinguishing between two cases:

Case1: The rule-set chooses the correct decision D_k for the i-th example.
Case1A: $P_{k,i} = \emptyset$ or $ON_{k,i} = \emptyset$
if $r \in P_{k,i}$ then $f'(tr_i, r) = \frac{\lambda_{r,i}}{Mult(P_{k,i})}$
else
if $r \in ON_{k,i}$ then $f'(tr_i, r) = \frac{\frac{1}{\lambda_{r,i}}}{Mult(ON_{k,i})}$
else 0
Case1B: $P_{k,i}$ and $ON_{k,i}$ are both nonempty
if $r \in P_{k,i}$ then $f'(tr_i, r) = \frac{p_{k,i}}{p_{k,i}+on_{k,i}} * \frac{\lambda_{r,i}}{Mult(P_{k,i})}$
else
if $r \in ON_{k,i}$ then $f'(tr_i, r) = \frac{on_{k,i}}{p_{k,i}+on_{k,i}} * \frac{\frac{1}{\lambda_{r,i}}}{Mult(ON_{k,i})}$
else 0

Case2: The rule-set chooses decision D_j incorrectly, with D_k being the correct decision for the i-th example.

Case2A: $P_{j,i} = \emptyset$ or $N_{k,i} = \emptyset$
if $r \in P_{j,i}$ then $f'(tr_i, r) = -\frac{\lambda_{r,i}}{Mult(P_{j,i})}$,
else
if $r \in N_{k,i}$ then $f'(tr_i, r) = -\frac{\frac{1}{\lambda_{r,i}}}{Mult(N_{k,i})}$,
else 0

Case2B : $P_{j,i}$ and $N_{k,i}$ are both non-empty
if $r \in P_{j,i}$ then $f'(tr_i, r) = -\frac{p_{j,i}}{p_{j,i}+n_{k,i}} * \frac{\lambda_{r,i}}{Mult(P_{j,i})}$,
else
if $r \in N_{k,i}$ then $f'(tr_i, r) = -\frac{n_{k,i}}{p_{j,i}+n_{k,i}} * \frac{\frac{1}{\lambda_{r,i}}}{Mult(N_{k,i})}$,
else 0.

To illustrate how this approach works, let us assume that we have to evaluate the IRIS-flower rule-set for the three examples given in Fig. 1. For each test-case, all five rules are fired, producing an odds multiplier each. Then, based on these multipliers the answer of the rule-set is computed and reward or punishment is applied. Table 1 depicts the necessary information for each test-case: the class to which the example belongs, the odds-multiplier of each rule of the rule-set, the answer of the rule-set for it. Moreover, Table 1 gives the reward each rule obtained for a particular test-case.

	Example 1	Example 2	Example 3	Σ Reward
Class	*Setosa*	*Versicolor*	*Virginica*	–
R1-mult	5.0	1.64	0.2	–
R2-mult	0.1	8.43	0.1	–
R3-mult	5.5	5.5	0.8	–
R4-mult	1.45	0.2	2.59	–
R5-mult	1.27	1.27	3.61	–
Decision	*Setosa*	*Setosa*	*Virginica*	–
R1-reward	0.32	-0.23	0.18	0.27
R2-reward	0.33	0.0	0.37	0.7
R3-reward	0.35	-0.77	0.05	-0.37
R4-reward	0.0	0.0	0.17	0.17
R5-reward	0.0	0.0	0.23	0.23

Table 1: Reward Computations for Fig. 1

Let us illustrate how these rewards have been computed for the three example. The first example (0.3, 0.0, 1.0, 0.67, Setosa) has been classified correctly by the rule-set as a Setosa; consequently, rules that provide positive evidence for Setosa (in the particular case, R1 and R3 provide positive evidence for Setosa), and negative evidence for other decision candidates (in the particular case, R2 provides negative evidence for Versicolor) are rewarded, whereas rules R4 and R5 are not included in the rewarding process, because they provided positive evidence for an incorrect decision — Virginica in the particular case.

This raises the question what total amount of rewards should be allocated to the three rules, and how rewards are distributed between R1, R2, and R3. Our approach, assigns an reward of +1 to a test-case that has been correctly, and of -1 to test-cases that have been classified incorrectly. In this particular case two groups of rules are rewarded: Group1: those that provide positive evidence for Setosa, Group2: those that provide negative evidence for Virginica or for Versicolor. If rules belonging to multiple rule-groups have to be rewarded, rewards are divided based on rule-group cardinalities. In the particular example, Group1 contains two rules, whereas group Group2 contains a single rule. Consequently, rules belonging to the first group obtain a total reward of 2/3, and the single rule belonging to the second group gets a reward of 1/3. Rewards within a group are subdivided proportionally to their odds multipliers. In the particular case R1's odds-multiplier of 5.0 is slightly lower than R3's odds multiplier of 5.5, consequently R3 receives a slightly higher portion of the reward. Figure 3 depicts the reward computations for Example 1, Example 2, and Example 3.

```
f'(example1,R1) = 2/(1 + 2)  * 5.0/(5.0 + 5.5) = 0.32
f'(example1,R2) = 1/(1 + 2)  * 0.1/0.1 = 0.33
f'(example1,R3) = 2/(1 + 2)  * 5.5/(5.0 + 5.5) = 0.35

f'(example2,R1) = -1 * 1.64/(1.64 + 5.5) = -0.23
f'(example2,R3) = -1 * 5.5/(1.64 + 5.5) = -0.77

f'(example3,R1) = 3/(2 + 3) * (1/0.2)/((1/0.2)+(1/0.1)+(1/0.8)) = 0.18
f'(example3,R2) = 3/(2 + 3) * (1/0.1)/((1/0.2)+(1/0.1)+(1/0.8)) = 0.37
f'(example3,R3) = 3/(2 + 3) * (1/0.8)/((1/0.2)+(1/0.1)+(1/0.8)) = 0.05
f'(example3,R4) = 2/(2 + 3) * 2.59/(2.59 + 3.61) = 0.17
f'(example3,R5) = 2/(2 + 3) * 3.61/(2.59 + 3.61) = 0.23
```

Figure 3 : Reward Distribution for the 3 Examples

This rewarding process continues until all examples of the training set have been processed. In our particular case, after evaluating three cases, it looks that R2 is a good rule, R1, R4, and R5 are decent rules, whereas R3 performed badly.

In summary, the reward punishment mechanism that was introduced in this section takes a training set and a rule-set as an input, and returns the score for each rule, measuring the classification performance of the rule for the training set.

3.3 Benefits of RPM

In the DELVAUX learning environment we use RPM to reduce the randomness of the genetic algorithm search process. For example, we developed a new version of DELVAUX that employs a mutation operator that uses the roulette wheel method based on a rule's score, which replaces rules with low scores with a higher probability than rules with high score, making the search process more directed. We compared this version with the basic version for a larger machine learning benchmark consisting of 20 different training-set/test-set pairs[TO93]. Using RPM improved the training performance by 1.8% and the testing performance by 0.76%. Moreover, we developed another version that combined RPM with heuristics that generate new rules more intelligently. In this version, RPM improved the testing performance for the benchmark more significantly by 1.7%.

It should be stressed that RPM is also be useful for the testing and debugging of rule-based systems that have been written by human beings. For example, RPM can help to find rules that were responsible for a fault more quickly, saving debugging time. Moreover, RPM can be used for the fine-tuning of rules by helping to identify rules that need minor adjustments, which is a very time consuming process when developing rule-based systems that use approximate reasoning techniques (see also [EI93]).

4 Summary

The main contribution of the paper is the reward punishment mechanism (RPM), which is an evaluation scheme for rule-based systems that rely on decision making by evidence combination. In contrast to the bucket brigade algorithm, RPM is a fuzzy evaluation scheme: rules are rewarded and punished proportionally to their contribution to the decision making process. We also claim that our approach is unique in the sense that it evaluates rules as well as rule-sets in the learning process.

5 References

[DJ90] De Jong, K.: "Genetic Algorithm-based Learning" in Kodratoff (Eds.) et al.: "Machine Learning", Volume 3, Chapter 21, Morgan Kaufmann, San Mateo, 1990.

[EI93] Eick, C. and Mehta, N.: " Decision Making Involving Imperfect Knowledge", IEEE Transactions on Systems, Man, and Cybernetics, pp. 840-850, June 1993.

[EJ93] Eick, C. and Jong, D.: "Learning Bayesian Classification Rules Through Genetic Algorithms", in Proc. Int. Conf. on Information and Knowledge Management, pp. 305-313, Washington, Nov. 1993.

[HO86] Holland, J.: "Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems", in Michalski,R. et al.(Eds.): Machine Learning, an Arficial Intelligence Approach (Vol.2), Morgan Kaufmann, 1986.

[JA93] Janikow, Cezary: "A Knowledge-Intensive Genetic Algorithm for Supervised Learning", Machine Learning, vol. 13, pp. 189-228, 1993.

[OL93] Oliver,Jim: "Discovering Individual Decision Rules: an Application of Genetic Algorithms", in Proc. Fifth Int. Conference on Genetic Algorithms, pp. 216-222, Chicago, October 1993.

[TO93] Toto, Ema: "Combining Rule Debugging Techniques with Genetic Algorithms — a Hybrid Approach to Learn Rules for Classification Tasks", Master's Thesis, Department of Computer Science, University of Houston, December 1993.

Traps and Pitfalls when Learning Logical Definitions from Relations

Floriana Esposito, Donato Malerba and Giovanni Semeraro

Laboratorio di Acquisizione della Conoscenza e Apprendimento nelle Macchine
Dipartimento di Informatica, Università degli Studi di Bari
Via Orabona 4, 70126 Bari, ITALY
{esposito, malerbad, semeraro}@vm.csata.it

Clifford Brunk and Michael Pazzani
Department of Information and Computer Science, University of California, Irvine
Irvine, CA 92717 USA
{brunk, pazzani}@ics.uci.edu

Abstract. In the paper, we present some learning tasks that cannot be solved by two well-known systems, FOIL and FOCL. Two kinds of explanations can be provided for these failures. For some tasks, the failures can be ascribed to a wrong definition of the space in which these systems perform the search for logical definitions. By moving from θ-subsumption to a weaker, but more mechanizable and manageable, model of generalization, called θ_{OI}-subsumption, a new search space is defined in which such tasks can be solved. Such a solution has been implemented in a new version of FOCL, called FOCL-OI. However, other learning tasks cannot be solved by changing the search space. For these tasks, the conceptual problem detected both in FOIL and in FOCL concerns the generation of meaningless rules, which do not mirror at all the structure of the training instances. We claim that, whenever possible, the training/test examples should be represented as ground Horn clauses, rather than as tuples of a relational database or facts of a Prolog database.

1 Introduction

Recently, Quinlan [20] has developed a system, called FOIL (First Order Inductive Learner), that learns logical definitions from data expressed as relations. FOIL proved effective and efficient on several learning tasks. Nevertheless, Quinlan himself recognizes that *"it is not difficult to construct tasks on which the current version of FOIL will fail"*. He ascribes these failures to the greedy search performed by FOIL. FOCL (First Order Combined Learner) [18] is an extension of FOIL in several aspects. The main extension allows FOCL to define and exploit background knowledge in the inductive learning process. Both FOIL and FOCL are widely acknowledged as an advance in the area of learning from examples [30] and some authors recently provided FOIL with a theoretical foundation by casting it in the Plotkin's framework [1].

In this paper, we point out some conceptual traps and pitfalls in which both FOIL and FOCL fall when they cope with toy world problems taken from the machine learning literature. The same problems occur on real world tasks [5, 14, 24]. Furthermore, we provide an interpretation of the behaviour of these systems based on a pure analytical approach to the problem of learning logical definitions. Section 2 presents a brief overview of FOIL and FOCL and describes the representation language used by these two systems. Negative experimental results concerning FOIL and FOCL are shown in Section 3. Moreover, we provide explanations for these negative results. These explanations straightforwardly suggest a way for overcoming the detected conceptual problems. The general theoretically-founded solutions are proposed in Section 4. In the same section, we present FOCL-OI, a new learning system based on FOCL, that replaces the model of generalization based on θ-subsumption [19] with that based on θ_{OI}-subsumption [7, 24]. FOCL-OI has been empirically evaluated on those learning tasks that could not be solved by FOIL and FOCL and proved successful in avoiding the problem of non-terminating learning processes.

2 FOIL and FOCL

Subsequently, we refer to [13] for what concerns the basic definitions of *substitution*, *positive* and *negative literal*, *clause*, *Horn clause* and *definite clause*. In particular, we denote a substitution (or *variable binding*) σ with $\{ x_1 \leftarrow t_1, x_2 \leftarrow t_2, \ldots, x_n \leftarrow t_n \}$, where the terms $t_1, t_2, \ldots, t_n$ replace the variables $x_1, x_2, \ldots, x_n$, respectively. We will consider a clause as the set of its literals. Henceforth, we denote with $\mathbb{L}$ the set of the *linked* Horn clauses [11]. The term *logic theory* is used as a synonym of *logic program*, i.e. a set of definite Horn clauses. Moreover, we refer to [19] for the definition of *θ-subsumption*.

θ-subsumption induces a *quasi-ordering* ≤[1] upon the set of the linked Horn clauses, that is, ≤ is reflexive and transitive, but not antisymmetric. Here, ~ denotes the equivalence relation induced by ≤. Notice that it does not coincide with set equality. Indeed, two equivalent clauses under ~ can be not only alphabetic variants, but even have a different number of literals, as for *{P(x), P(f(y))}* and *{P(f(z))}*. Thus, as Plotkin [19] pointed out, there is a *reduced member* of any equivalence class under ~ and this member is unique to within an alphabetic variant.

At the high level, both FOIL and FOCL adopt a *separate-and-conquer* search strategy. The *separate* stage of the algorithm is basically a loop that checks for the completeness of the current logical definition and, if this check fails, begins the search for a new consistent clause, while the *conquer* stage performs a general-to-specific hill-climbing search to construct the body of the new clause. The evaluation function used in the latter search is the information theoretic heuristic called *information gain*. Therefore, the search space for the separate stage is $2^{\mathbb{L}/\sim}$, i.e. the space of logic theories. The conquer stage searches for a consistent clause in the only specialization hierarchy of $(\mathbb{L}/\sim, \leq)$ rooted into the linked Horn clause whose head contains the predicate to be learned and whose body is empty, i.e. $P(X) \leftarrow$, where P is the predicate that denotes the concept to be learned. Indeed, both FOIL and FOCL adopt θ-subsumption as model of generalization.[2]

In FOIL, each k-ary predicate is associated with a *relation* consisting of the set of k-tuples of constants that satisfy that predicate. Such predicates are called *extensional* or *operational*. The relation associated to the predicate that we want to learn is called *target relation*. Each relation, including the target relation, defines both *positive* and *negative instances* of the predicate. A peculiarity of FOCL is that it is possible to define and use *intensional* or *non-operational* predicates, besides extensional ones. A predicate is defined intensionally when it appears in the head of an inference rule expressed as a Horn clause.

[1] Differently from Plotkin, we write C ≤ D to indicate that D is more general than C.

[2] This is true when FOCL does not use background knowledge.

Given a *k*-ary predicate *P*, FOIL and FOCL learn a *logical definition* or *rule* for *P*, that is, a set of definite Horn clauses in which $P(X_1, X_2, ..., X_k)$ is the literal in the head. Each clause of this logical definition should be satisfied only by positive tuples (*consistency*), while all clauses together should cover all positive instances of $P(X_1, X_2, ..., X_k)$ (*completeness*). Each literal in the body of a clause takes one of the four forms $X_j = X_k$, $X_j \neq X_k$, $Q(V_1, V_2, ..., V_r)$, $\neg Q(V_1, V_2, ..., V_r)$, where the X_i's are variables already introduced by previous literals (*old* variables) in the body, the V_i's can be both old and *new* variables, and *Q* is the same predicate associated with a relation. Therefore, FOIL and FOCL learn *function-free* Horn clauses, since terms other than variables cannot occur as arguments of a literal.

3 Experiments with FOIL and FOCL

This section presents some experimental results obtained by running FOIL and FOCL[3] on classical tasks from the machine learning literature.

3.1 Learning the concept of a bicycle

Helft [11] reports a very simple learning task. The learning system is given two descriptions of a bicycle:

B_1: bicycle(obj1) :- wheel(obj1, p1), wheel(obj1, p2)
B_2: bicycle(obj2) :- wheel(obj2, p3), wheel(obj2, p4)

Helft observes that the best generalization *á la* Plotkin obtainable from these examples is: G: bicycle(X) :- wheel(X, Y)
since the clause: G': bicycle(X) :- wheel(X, Y), wheel(X, Z)
is logically equivalent to *G* under θ-subsumption. As a consequence, the number of wheels does not appear in the logical definition of bicycle, even though it is a relevant characteristic.

As expected, both FOIL and FOCL learn the same rule *bicycle(X)* ← , when they cope with this task. This result is easily explained by the fact that these two systems successfully adopt a *lazy* learning strategy that biases the search towards the generation of maximally general discriminant descriptions rather than *least general generalizations* [16]. Therefore, when no negative examples are provided, the rule discovered will always consist of a single clause whose body is empty. The behaviour of these two systems is more interesting on three slightly different learning tasks, that we created purposely. The three learning problems differ from each other by the description of a unique negative example that we added to the original training set.

In the first problem, the training set consists of the positive examples B_1 and B_2 and the negative example N_1: N_1: non-bicycle(obj3) :-

Both FOIL and FOCL generate the same clause:

B: bicycle(A) :- wheel(A, B)

In the second problem, the only training example, N_2, is represented by the following clause: N_2: non-bicycle(obj3) :- wheel(obj3, p5)

In this case, FOIL and FOCL show a different behaviour. In fact, FOIL converges to

[3] All the experiments were run using FOIL5.0 [22] and FOCL-1-2-3 Version 1.1 [17].

the following concept definition:

B: bicycle(A) :- wheel(A, B), wheel(A, C), B<>C

while FOCL does not converge, that is, FOCL's search does not terminate. An analysis of the ongoing running provides a simple explanation of this behaviour. A snapshot of the state of FOCL's search reveals that FOCL is testing the following clause:

bicycle(?0) :- wheel(?0, ?1),wheel(?0, ?2),wheel(?0, ?3),wheel(?0, ?4),wheel(?0, ?5), ...

Notice that the addition of a literal *wheel(?0, ?n)*, $n = 2, 3, \ldots,$ to this partially developed clause does not change the equivalence class. Indeed, it is easy to see that the partial clause is logically equivalent under θ-subsumption to the clause:

bicycle(?0) :- wheel(?0, ?1)

Therefore, *FOCL's conquer stage searches in the wrong specialization hierarchy. It performs its search in the lattice* $(\mathbb{L}_h, \leq)$ *of all the linked Horn clauses rather than in the lattice* $(\mathbb{L}_h/\sim, \leq)$ *of all the equivalence classes* having a linked Horn clause as a reduced member *(1st trap)*.

FOCL does not implement an algorithm for testing the equivalence under θ-subsumption of the newly generated hypothesis with respect to the previous one. As a consequence, the search continues by generating and testing hypotheses that belong to the same equivalence class as the previously generated hypothesis, so remaining inside the same node of the specialization hierarchy and with no mean to get out from there. In other words, the conquer stage in FOCL defines incorrectly its termination condition, therefore there is no guarantee of termination.

The different behaviour of FOIL and FOCL is easily explained by the fact that FOIL, by default, extends the search also to the space of positive and negative literals in the form $X_j = X_k$ (see Section 2), while FOCL requires the user/teacher of the system explicitly set one of its built-in functions, namely the *eql* built-in function, that allows the system to test also the literals *eql(X, Y)* and their negations (alphabetic variants of $X = Y$ and $X \neq Y$, respectively), in order to do the same. Indeed, when such a parameter is set, FOCL learns the same rule discovered by FOIL. Note that, without introducing the inequality relationship between variables, no concept definition of bicycle consistent with respect to the given training examples exists in the search space.

In the last problem, the only training example, N_3, is represented by the following clause:

N_3: non-bicycle(obj3) :- wheel(obj3, p5), wheel(obj3, p6), wheel(obj3, p7)

In this case, FOIL is not able to learn any complete and consistent definition for the concept *bicycle*. It might be guessed that the problem with FOIL is that it adopts a *stopping criterion* for the conquer stage that is based on Rissanen's Minimum Description Length (MDL) principle [23]. But the elimination of this stopping criterion[4] has no visible effect. In fact, the analysis of the output reveals that the clause generated by FOIL's conquer stage is the following:

bicycle(A) :- wheel(A,B),
wheel(A,C), B=C,
wheel(A,D), B=D, ...

The addition of the pairs of literals *wheel(A,X), B=X* stops only because FOIL has an upper bound on the number of variables in a clause. In fact, the trace of the running reveals

[4] This change was made by modifying properly the C language macro `CostOfLit` in the file *defns.i* of FOCL5.0.

that even FOIL's conquer stage performs its search in $(\mathbb{L}, \leq)$ rather than in the lattice $(\mathbb{L}/\sim, \leq)$. Indeed, the literals *wheel(A,X)* are not only redundant, but even alphabetic variants of the existing literal *wheel(A,B)*.

3.2 Learning poker concepts

In a recent paper presenting a new learning system for multiple concept learning, named M-FOCL, Datta and Kibler [3] describe a problem which consists in learning a subset of the possible five card poker hands. Specifically, they learn the following concepts: *one pair*, *two pair*, *three of a kind*, *four of a kind* and *full house*. In the paper, the authors consider two alternative representations of the examples. In the former representation, an example consists of the 5-tuple *(card1, card2, card3, card4, card5)* and they use the predicates *rank(card, r)* and *suit(card, s)* to describe the rank and the suit respectively of each card in the 5-tuple. In the latter representation, the 1-tuple *(hand)* is used to describe an example. The predicates used in this representation are:

one-pair(H)	H is a hand containing exactly two cards of the same rank and whose suits are different
hand-contains-card(H, C)	C is a card of H
suit(C, S)	S is the suit of C
value(C, V)	V is the rank of C
value-succ(V_1, V_2)	V_2 is the successor of V_1
value<(V_1, V_2)	V_1 is less than V_2

Shortly, the main difference between the two representations is that the former introduces an artificial ordering among the cards in a hand, which causes a combinatorial explosion of the search [3, p. 91], whereas, in the latter, the relational nature of an example is represented more properly, since an example is one object (hand) which consists of five subparts (cards). However, both FOCL and M-FOCL have space limitation problems when learning the concepts with the latter representation. In particular, the reason for the reported failures lies in the fact that the unbound variable *?card* in each literal of the kind *hand-contains-card(?hand, ?card)* increases the search space by a factor of five. In addition to this, we found that FOCL, M-FOCL and FOIL may incur different problems when using such a representation. Indeed, we randomly generated a training set for learning the concept of *one-pair* consisting of four examples. Two of them are five card poker hands, containing exactly two cards of the same rank and different suits, and the remaining two examples are hands which do not contain any single pair (this means that they may contain three or four cards of the same rank and different suits or two cards of the same rank and suit).

FOIL takes .2 seconds to generate no clause, as expected, while FOCL does not converge at all. The partial clause generated (Figure 1) shows clearly that FOCL will not converge to any complete and consistent rule. Furthermore, this time, besides performing its search within the same equivalence class of the specialization hierarchy of $(\mathbb{L}/\sim, \leq)$ rooted into the linked Horn clause *one-pair(H)* ←, *FOCL is actually exploring a set of completely meaningless clauses*, as that shown in Figure 1, whose English interpretation is: *"?0 is a pair if there exist 6 hands which contain the same card (?1)."*

Even more so, the property described by this rule occurs in no training examples given to the learning system (*2nd trap*). This happens because FOCL, as well as FOIL, generates and tests all the possible *variabilizations* of the defined predicates, provided that they do not violate the constraint of linkedness of the current clause, that is, all the variabilizations that contain at least one old variable. Among all the clauses obtained by combining all the possible variabilizations of the predicates, some of them do not match at all the structure of the training examples. Nonetheless, they are considered promising by the information gain heuristic because the original training examples do not exist anymore as a whole, i.e. as *ground* definite clauses, since they have been *atomized* and stored as tuples of a relational database, according to Quinlan's paradigm [20] for "Learning Logical Definitions from Relations". In particular, problems raise when some relations are *reified* in a *conceptualization* of the world [9].

The following example should make clear the limits of an over simplistic use of a relational database to represent the training instances in a system that learns logical definitions.

Example. Consider the following two five card poker hands, each one containing an instance of the concept *one-pair* and represented as:

one-pair(H$_1$) :- hand-contains-card(H$_1$, J-H), hand-contains-card(H$_1$, J-S),
hand-contains-card(H$_1$, 9-H), hand-contains-card(H$_1$, 8-H),
hand-contains-card(H$_1$, A-S), ...
one-pair(H$_2$) :- hand-contains-card(H$_2$, A-S), hand-contains-card(H$_2$, K-H),
hand-contains-card(H$_2$, K-S), hand-contains-card(H$_2$, 10-D),
hand-contains-card(H$_2$, 7-H), ...

where, for clarity sake, we neglected to report all the literals whose predicate is different from *hand-contains-card*. Both in FOIL and in FOCL these two examples are represented as the following set of tuples of the relation *hand-contains-card*:

hand-contains-card = { <**H**$_1$, J-H>, <**H**$_1$, J-S>, <**H**$_1$, 9-H>, <**H**$_1$, 8-H>, <**H**$_1$, **A-S**>,
<**H**$_2$, **A-S**>, <**H**$_2$, K-H>, <**H**$_2$, K-S>, <**H**$_2$, 10-D>, <**H**$_2$, 7-H> }

When any of these two learning systems performs its search in the hypothesis space, it is possible that it generates a meaningless definition for *one-pair*, as that in Figure 1, since the only *tools* available for the search are the linkedness and the information gain heuristic. Each linked clause defines a *thread* of arguments (in bold for the relation *hand-contains-card*) within the relational database, that joins all the tuples directly or indirectly related to the arguments in the head of the clause. Notice that, if we pull up the thread from the head H_1, all the tuples in the *hand-contains-card* relation hang from the thread, because of the ace of spades *(A-S)*. This is due to the reification of the relation *A-S*.

The example above demonstrates that linkedness alone cannot prevent a first-order learner, which uses a relational database to store its training instances, from losing the structure of an example as a whole and, consequently, from generating and testing all the possible variabilizations of the given primitives, including those that give rise to completely meaningless clauses, which do not mirror the structure of any training

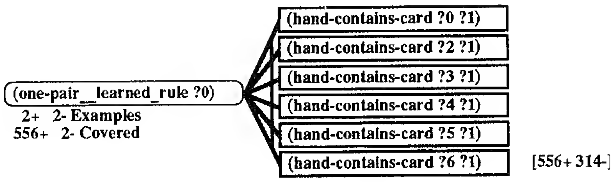

(one-pair__learned_rule ?0)
2+ 2- Examples
556+ 2- Covered
(hand-contains-card ?0 ?1)
(hand-contains-card ?2 ?1)
(hand-contains-card ?3 ?1)
(hand-contains-card ?4 ?1)
(hand-contains-card ?5 ?1)
(hand-contains-card ?6 ?1)
[556+ 314-]

Figure 1. The inconsistent partial clause produced by FOCL for the concept *one-pair* in the *poker* domain.

instance. The knowledge representation consideration underlying the conclusion above also applies to those learning systems from the area of Inductive Logic Programming (ILP) that store the training examples as ground unit clauses (*facts*) in a Prolog database, by splitting the body of an example into its composing literals. To sum up, whenever possible, it is necessary to store the training instances as ground Horn clauses in a data structure that preserves the intrinsically relational nature of the examples. Then, the positive and negative examples covered by a clause, as well as the information gain of each candidate literal, should be computed with respect to this structure, rather than with respect to the tuples in the relational database.

As to FOIL, the previously reported problem seems to be due to the MDL-based stopping criterion. However, by turning it off, the rule generated by FOIL contains a clause which describes a pattern that occurs in no positive training example, if we consider the five card hands as our training examples.

4 Theoretical solutions

The theoretical solution to the first kind of trap is to change the search space of the conquer stage of FOIL and FOCL from $(\mathfrak{L}, \leq)$ to $(\mathfrak{L}/\sim, \leq)$.

Practically speaking, this means to provide the conquer stage of the high level separate-and-conquer strategy with a procedure that performs a θ-subsumption equivalence test. This procedure should check if the current clause is equivalent (under ~) to the clause obtained by adding the literal with the highest information gain. In the general case, this is an NP-complete problem, since it involves a test for θ-subsumption, which is NP-complete [8]. In the particular case of FOIL and FOCL, the current clause *C* is a proper subset of the clause *D*, obtained by adding the literal with the maximum information gain, since the conquer stage performs a general-to-specific search. Thus, the procedure needs to check only if *D* θ-subsumes *C*. In formulae, $C \subset D$ and we want to know if $C \sim D$. But $C \subset D$ implies that $D \leq C$, thus it is enough to check whether $C \leq D$.

However, let us observe that, for the conquer stage of FOIL and FOCL, the following proposition holds (the proof can be found in [7]):

Proposition *Let C and D be two definite Horn clauses:*

$$C \textit{ reduced}, C \subset D \textit{ and } vars(C) = vars(D) \Rightarrow [C]_{\sim} \neq [D]_{\sim}$$

where $vars(\phi)$ is the set of the variables occurring in the clause ϕ. In other words, it is necessary to perform a θ-subsumption equivalence test if and only if the new clause *D* introduces new variables. Therefore, since in FOIL and FOCL we have $D = C \cup \{L_{n+1}\}$, where L_{n+1} is the literal with maximum information gain, and *C* and *D* are linked, it is sufficient to search for a unification between L_{n+1} and one of the *n* literals in *C*. This search has a linear complexity in the size of the clause. Indeed, it can be proved that there exists no ascending chain of infinite length $\{C_i\}_{i\geq 0}$ such that $C_i \leq C_{i+1}$ and $C_i \subset C_{i+1}$. The trivial case in which $L_{n+1} \in C$ can be easily ruled out by preventing the search from adding a literal that already exists in the body of the currently generated clause.

As a consequence, the computational complexity of the procedure that performs a θ-subsumption equivalence test can be reduced to $O(n)$, where *n* is the *size* of the current clause *C*. Indeed, from the literature about unification [12], it is known that the best unification algorithms between terms have a linear complexity in time.

For the sake of completeness, we have to observe that a literal which appears to be redundant in a partially developed clause (*local redundancy*) may no longer be such when further literals are added to the clause. There are two possible solutions to this short-sightedness of the search. A simple one consists in testing pairs (triples, foursomes, ...) of literals, when there is no single literal that helps to jump to a different equivalence class, while the other solution consists in testing pairs of literals $\{L_{n+1}, L_{n+2}\}$ as soon as L_{n+1} turns out to be redundant.

The equivalence test is no longer necessary when we adopt a model of generalization weaker than θ-subsumption, but more mechanizable and manageable, called θ-subsumption under object identity (θ_{OI}-subsumption) [24, 7].

Definition (θ_{OI}-subsumption) Let C, D be two Horn clauses. We say that D *θ-subsumes C under object identity* (D *θ_{OI}-subsumes* C) iff there exists a substitution σ such that $D\sigma \subseteq C$ and *σ is injective.*

Definition (Generality under object identity) Let C, D be two Horn clauses. We say that D is *more general than or equal to C under object identity* and we write $C \leq_{OI} D$ iff D θ_{OI}-subsumes C, that is, $C \leq_{OI} D$ iff $\exists\, \sigma : D\sigma \subseteq C$ and σ is injective

The change of the underlying model of generalization from θ-subsumption to θ_{OI}-subsumption causes a change of FOCL's search space from $(\mathfrak{L}, \leq)$ to $(\mathfrak{L}, \leq_{OI})$. Indeed, it is easy to see that the proposition above holds even under object identity. Moreover, the introduction of a new variable always causes a change of the equivalence class under object identity. Thus, we can conclude that searching into $(\mathfrak{L}, \leq_{OI})$ is equivalent to searching into $(\mathfrak{L}/\sim, \leq_{OI})$. In order to avoid non-termination problems (1st kind of trap), we changed the underlying model of generalization of FOCL from θ-subsumption to θ_{OI}-subsumption. This new version of FOCL, called FOCL-OI, implements object identity in a modular way. Indeed, the end-user/teacher can choose the type(s) of variables for which object identity is assumed.

Figure 2 shows the logical definition of a *bicycle* generated by FOCL-OI when θ_{OI}-subsumption is assumed as generalization model for the variables whose type is *wheel*. Note that the rule produced is the same as that found by FOCL when the *eql* built-in function is set. Object identity assumption allows FOCL to prevent the problem of non-termination of the search performed by the conquer stage and to recover the relevant property concerning the number of wheels, with no user intervention (to set the *eql* built-in function).

The second trap, detected in the *poker* domain, is not solved by changing the search space from $(\mathfrak{L}, \leq)$ to either $(\mathfrak{L}/\sim, \leq)$ or $(\mathfrak{L}, \leq_{OI})$. In other words, the θ-subsumption equivalence test is not enough to avoid this kind of problem. It requires further processing. Specifically, it requires that the ILP system is able to reconstruct the training examples as ground Horn clauses (*patterns*) from the ground unit clauses (*facts*) existing in its database, before starting the learning process. In this phase of *pattern reconstruction*, the ILP system must exploit both language biases, such as linkedness, and search biases, such as *determinate literals* [21] or *relational clichés* [26]. When the patterns that represent the training instances cannot be reconstructed completely, the learning process may be irreparably jeopardized, as the results in the *poker* domain show.

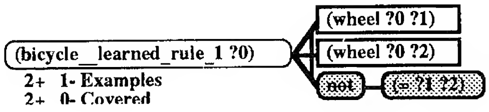


Figure 2. Complete and consistent definition for the concept *bicycle* produced by FOCL-OI. The shaded literals are the *inequality literals.*

Some existing learning systems avoid the second kind of trap by adopting different shrewdnesses. For instance, ML-SMART [2] uses a relational database to store the training instances, but each tuple in the database is actually a pointer to another structure which contains the original examples, represented as ground Horn clauses. The systems by Winston [29], Hayes-Roth [10], and Vere [28] represent the examples as graphs or equivalent formalisms (semantic networks, parameterized structural representations, conjunctions of literals, respectively). The family of INDUCE learning systems [15] and INDUBI [4, 6, 25] represent the examples as decision rules, a representation language equivalent to *colored* graphs [27]. Furthermore, they adopt the generalization model defined by the operation of subgraph isomorphism (equivalent to a restriction of θ_{OI}-subsumption to decision rules) to prevent the first kind of traps and the concept of *seed* to prevent the second kind of traps. A new system, called INDUBI/H [6, 25], represents the examples as ground definite clauses and fully adopts the model of generalization defined by θ-subsumption under object identity. INDUBI/H is an extension of INDUBI in several aspects. The main extension allows INDUBI/H to learn linked (definite) Horn clauses instead of decision rules, that can be viewed as Horn clauses with no head, i.e. variables are not allowed to occur in the action part of the rule.

As to FOIL and FOCL, the solution adopted by ML-SMART seems the simplest to be implemented among all those listed above, since all these three systems adopt the same internal representation of the examples.

5 Conclusions and future work

All learning systems need to clarify the adopted model of generalization and the space in which they perform the search for logical definitions. Formal methods and techniques can be useful to detect potential sources of problems and conceptual shortcomings of the existing empirical learning systems and to suggest the suitable counteractions in order to improve their performance.

This paper constitutes an attempt to apply this analytical approach to two well-known learning systems, FOIL and FOCL. Plotkin's logical framework to inductive generalization [19] proved useful both to point out some lacks that affect the search strategy of these systems and to suggest straightforwardly the adequate corrections. These corrections have been partially implemented in a new version of FOCL, called FOCL-OI, and proved effective to overcome some of the detected problems. Future work will concern the implementation of a new version of FOCL in which, at first, the θ_{OI}-subsumption generalization model is fully adopted (for any type of variables) and then, the system autonomously learns the *equality literals* in the form $[X_i=X_j]$, if necessary. Theoretically, this new learning strategy should prove more effective and efficient than that in which θ-subsumption and *difference links* $[X_i \neq X_j]$ have to be learned, since θ_{OI}-subsumption is more manageable than θ-subsumption, while learning the equality literals would happen rarely, according to a *when-needed* strategy.

References

1. Bell, S., and Weber, S., On the close logical relationship between FOIL and the frameworks of Helft and Plotkin, *Proceedings of The Third International Workshop on Inductive Logic Programming ILP'93*, Bled, Slovenia, 1-10, 1993.
2. Bergadano, F., Giordana, A., and Saitta, L., Automated concept acquisition in noisy environments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-10, 555-578, 1988.

3. Datta , P., and Kibler, D., Concept Sharing: A Means to Improve Multi-Concept Learning, *Proceedings of the 10th International Conference on Machine Learning*, Amherst, MA, 89-96, 1993.
4. Esposito, F., Automated acquisition of production rules by empirical supervised learning methods, in *Data, Expert Knowledge and Decisions*, (Vol. II), M. Schader (Ed.), Springer-Verlag, Heidelberg, Germany, 1990.
5. Esposito, F., Malerba, D., Semeraro, G., and Pazzani, M., A Machine Learning Approach To Document Understanding, *Proceedings of the 2nd International Workshop on Multistrategy Learning MSL-93*, Harpers Ferry, West Virginia, 276-292, 1993.
6. Esposito, F., Malerba, D., and Semeraro, G., Multistrategy Learning for Document Recognition, *Applied Artificial Intelligence* 8, 33-88, 1994.
7. Esposito F., Malerba, D., Semeraro, G., Brunk, C., and Pazzani, M., Avoiding Non-Termination when Learning Logical Programs: A Case Study with FOIL and FOCL, in *Pre-proceedings of LOPSTR 94 - Fourth Workshop on Logic Program Synthesis and Transformation*, Pisa, Italy, June 20-21, 1994.
8. Garey, M.R., and Johnson, D.S., *Computers and Intractability*, Freeman, San Francisco, CA, 1979.
9. Genesereth, M.R., and Nilsson, N.J., *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Palo Alto, CA, 1987.
10. Hayes-Roth, F., Schematic classification problems and their solution, *Pattern Recognition*, 105 - 113, 1974.
11. Helft, N., Inductive Generalization: A Logical Framework, in *Progress in Machine Learning - Proceedings of EWSL 87*, I. Bratko & N. Lavrac (Eds.), Sigma Press, Bled, Yugoslavia, 149-157, 1987.
12. Knight, K., Unification: A Multidisciplinary Survey, *ACM Computing Surveys*, Vol.21, No.1, 1989.
13. Lloyd, J.W., *Foundations of Logic Programming*, Second Edition, Springer-Verlag, New York, 1987.
14. Malerba, D., *Document Understanding: A Machine Learning Approach*, Technical Report, Esprit Project 5203 INTREPID, March 1993.
15. Michalski, R.S., Pattern Recognition as Rule-Guided Inductive Inference, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, 349-361, 1980.
16. Muggleton, S., Inductive Logic Programming, *New Generation Computing*, 8(4), 295-318, 1991.
17. Pazzani, M.J., and Brunk, C., *FOCL-1-2-3 Version 1.1*, Department of Information and Computer Science, University of California, Irvine, California, March 1993.
18. Pazzani, M., and Kibler, D., The utility of knowledge in inductive learning, *Machine Learning* 9, 1, 57-94, 1992.
19. Plotkin, G.D., A Note on Inductive Generalization, in *Machine Intelligence 5*, B. Meltzer and D. Michie (Eds.), 153-163, Edinburgh University Press, 1970.
20. Quinlan, J. R., Learning Logical Definitions from Relations, *Machine Learning* 5, 3, 239-266, 1990.
21. Quinlan, J. R., Determinate Literals in Inductive Logic Programming, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Sydney, Australia, 746-750, 1991.
22. Quinlan, J. R., Cameron-Jones, R.M., FOIL: a midterm report, in *Machine Learning: ECML-93 - Proceedings of the Sixth European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence 667, Pavel B. Brazdil (Ed.), Springer-Verlag, Vienna, Austria, 3-20, 1993.
23. Rissanen, J., A universal prior for integers and estimation by minimum description length, *Annals of Statistics*, 11, 1, 416-431, 1983.
24. Semeraro, G., Brunk, C.A., and Pazzani M.J., *Traps and Pitfalls when Learning Logical Theories: A Case Study with FOIL and FOCL*, Technical Report 93-33, Department of Information and Computer Science, University of California, Irvine, California, July 26, 1993.
25. Semeraro, G., Esposito, F., and Malerba, D., Learning Contextual Rules for Document Understanding, *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*, San Antonio, Texas, 108-115, 1994.
26. Silverstein, G., and Pazzani, M., Relational clichés: constraining constructive induction during relational learning, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, 203-207, 1991.
27. Stepp, R. E., *Conjunctive Conceptual Clustering: A Methodology and Experimentation*, Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois, UIUCDCS-R-84-1189, 1984.
28. Vere, S.A., Induction of relational productions in the presence of background information, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, MA, 349-355, 1977.
29. Winston, P.H., *Learning Structural Descriptions from Examples*, Ph.D. dissertation, Dept. of Electrical Engineering, Massachussetts Institute of Technology, Cambridge, MA, January 1970.
30. Wirth, R., and O'Rorke, P., Constraints on Predicate Invention, *Proceedings of the Eighth International Workshop on Machine Learning*, Evanston, Illinois, 457-461, 1991.

DBROUGH: A Rough Set Based Knowledge Discovery System *

Xiaohua Hu, Ning Shan, Nick Cercone, Wojciech Ziarko

Department of Computer Science, University of Regina
Regina, SK, Canada, S4S 0A2
e-mail: {xiaohua, ning, nick, ziarko}@cs.uregina.ca

Abstract. Knowledge discovery in databases, or data mining, is an important objective in the development of data- and knowledge-base systems. An attribute- oriented rough set method is developed for knowledge discovery in databases. The method integrates learning from example techniques with rough set theory. An attribute-oriented concept tree ascension technique is first applied in generalization, which substantially reduces the computational complexity of the database learning processes. Then the rough set techniques are applied to the generalized relation to derive different knowledge rules. Moreover, the approach can find all the maximal generalized rules in the data. Based on these principles, a prototype database learning system, DBROUGH, has been constructed. Our study shows that attribute-oriented induction combined with rough set techniques provide an efficient and effective mechanism for knowledge discovery in database systems.

Keywords: Machine Learning, Knowledge Discovery in Databases, Methodologies

1 Introduction

Learning is one of the most important characteristics of human and machine intelligence. Since relational database systems are pervasive and widely utilized in many applications, it is advantageous to learn knowledge rules from data in relational databases. Recently, data mining has been ranked as one of the most promising research topics for the 1990s by both database and machine learning researchers [10].

In our previous study [1,5,6], an attribute-oriented induction method was developed. Our method integrates *learning from example* techniques with database operations and extracts generalized data from actual data in the databases.

We further develop the attribute-oriented approach by integrating rough set theory [8]. We expand our system further for learning decision rules, characteristic rules and discrimination rules. Moreover, our system can find all the maximal generalized rules.

* The authors are members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centers of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc.

2 Principal Concepts of Rough Set

Rough set was first introduced in [8]. In this section we introduce some of the principal notions of rough set and present a new method to find the best reduct or a minimal attribute subset which meets the user's requirement.

2.1 Information System

By an information system S, we mean that $S = \{U, A, V\}$, where U is a finite set of object, $U = \{x_1, x_2, ..., x_n\}$, A is a finite set of attributes, the attributes in A is further classified into two disjoint subsets, *condition* attributes C and *decision* attributes D. $V = \bigcup_{p \in A} V_p$ and V_p is a *domain* of attribute p.

let $P \subset A$, $x_i, x_j \in$ U. we define a binary relation $\tilde{P}$, called an *indiscernibility relation*, as $\tilde{P} = \{(x_i, x_j) \in U \times U : for\ every\ p \in P\ p(x_i) = p(x_j)\}$. We say that x_i and x_j are indiscernible by set of attributes P in S iff $p(x_i) = p(x_j)$ for every $p \in P$. One can check that $\tilde{P}$ is an equivalence relation on U for every $P \subset A$. Equivalence classes of relation are called P-elementary sets in S. A-elementary sets are called atoms of S. Information system S is selective iff all atoms in S are one element set, i.e. A is an identity relation.

A relational database may be considered as an information system in which columns are labeled by attributes, rows are labeled by the objects and the entry in column p and row x has the value $p(x)$. Each row in the relational table represents *information* about some object in U. The difference is that the entities of the information systems do not need to be distinguished by their *attributes* or by their relationship to entities of another type. In the relational database, one attribute is identified as a *decision* attribute, and the other attributes are the *condition* attributes. In this paper, we adopt the view that a relational database is a selective information system and will use the term *relational database* and *information system* interchangeably.

2.2 Approximation Space

For the information system $S = \{U, A, V\}$, and $IND \subset A$ is an equivalence relation (*indiscernibility relation*) on U, an order pair $AS = (U, IND)$ is called an *approximation space*. For any element x_i of U, the equivalence class of x_i in relation IND is represented as $[x_i]_{IND}$. Equivalence classes of IND are called *elementary sets in* AS because they represent the smallest discernible groups of objects. Any finite union of elementary sets in AS is called a *definable set in* AS.

Let $X \subset U$, we want to define X in terms of *definable sets* in AS, thus we need to introduce the following notions cited from [8]:

(i) The lower approximation of X in AS is defined as $\underline{IND}X = \{x_i \in U | [x_i]_{IND} \subset X\}$. $\underline{IND}$X is the union of all those elementary sets each of which is contained by X. For any $x_i \in \underline{IND}$X, it is certain that it belongs to X.

(ii) The upper approximation of X in AS is defined as $\overline{IND}X = \{x_i \in U | [x_i]_{IND} \cap X \neq \emptyset\}$. $\overline{IND}$X is the union of those elementary sets each of which has a non-empty intersection with X. For any $x_i \in \overline{IND}$X, we can only say that x_i is possible to belong to X.

(iii) The set $\overline{IND}$X-$\underline{IND}$X is called the IND-doubtful region of IND in (U, IND). For any $x_i \in$ U, if x_i in $\overline{IND}$X-$\underline{IND}$X, it is impossible to determine that x_i belong to X based on the descriptions of the elementary sets of IND.

2.3 Core and Reducts of Attributes

Core and reduct are the two fundamental concepts of rough set. A reduct is the essential part of an information system which can discern all objects discernible by the original information system. A core is the common parts of all the reducts.

Let $S = \{U, A, V\}$ be an information system, $A = C \cup D$, B $\subset$ C, we define a positive region B in $IND(D)$, $POS_B(D)$, as $POS_B(D) = \cup\{\underline{B}X : X \in IND(D)\}$. The positive region $POS_B(D)$ includes all objects in U which can be classified into classes of $IND(D)$ without error just based on the classification information in $IND(B)$.

We say that the set of attributes D depends in degree k $(0 \leq k \leq 1)$ on the subset R of C in S if $k(R, D) = card(POS_R(D))/card(POS_C(D))$. The value $k(R, D)$ provides a measure of dependency between R and D. If R is a reduct of C with respect with D, then $k(R, D) = 1$.

The significance of an individual attribute a added to the set R with respect to the dependency between R and D is represented by significant factor SGF, given by $SGF(a, R, D) = k(R + \{a\}, D) - k(R, D)$.

$SGF(a, R, D)$ reflects the degree of increase of dependency level between R and D as a result of the addition of the attribute a to R. In practice, the stronger the influence of the attribute a is on the relationship between R and D, the higher the value of the $SGF(a, R, D)$ is.

Definition 2.1. An attribute $p \in B$ is superfluous in B with respect to D if $POS_B(D) = POS_{B-\{p\}}(D)$, otherwise p is indispensable in B with respect to D.

If an attribute is superfluous in the information system, it can be removed from the information system without changing the dependency relationship of the original system.

Definition 2.2. If every attribute of B is indispensable with respect to D, then B is indispensable with respect to D.

Definition 2.3. The set of all indispensable attributes in C with respect to D is called the core of C and is denoted by $CORE(C, D)$.

$CORE(C, D) = \{a \in C : POS_C(D) \neq POS_{C-\{a\}}(D)\}$.

The concept of the core can be used as the starting point for computation of reducts. Some methods for computing the core of the information system are provided in [11,12].

Definition 2.4. $B \subset C$ is defined as reduct in S if B is independent with respect to D and $POS_C(D) = POS_B(D)$

The reduct of C is a minimal subset of attributes that discerns all object discernible by the whole set of attributes. Usually, C may have more than one reduct. We adopt the criteria that the best reduct is the one which has the minimal number of attributes and if there are two or more reducts with same minimal number of attributes, then the reduct with the least number of combinations of values of its attributes is selected.

Algorithm 1 (Reduct Algorithm:) *Compute the best reduct or user minimal attribute subset.*

Input: (i) The task-relevant generalized relation R' (ii) a set of attributes AR for relation R', which is classified into condition attributes C, and decision attributes D (iii) the core CO of AR computed. (CO may be empty) (iv) the attribute set UA user prefer to emphasize (UA may be empty, if UA is empty, that means the user does not have preference for any attribute)
Output. A set of attributes $REDU$
Method

```
Step 1: $REDU = CO \cup UA$;
Step 2: $AR = AR - REDU$
Step 3: Find attribute $a$ in $AR$ which has the maximal value $SGF(a, REDU, D)$.
Step 4: If there are several attributes $a_i$ (i=1,...,m) with the same maximal value $SGF(a, REDU, D)$, choose the attribute $a_j$ which has the least number of combination value with $REDU$.
Step 5: $REDU = REDU \cup \{a_j\}$, $AR = AR - \{a_i\}$ (i=1,...,m)
Step 6: If $K(REDU, D)$ =1, then terminate, otherwise go to Step 3.
```

We can find the best reduct or user minimal attribute subset in $N_A \times O(N' \times N')$ in the worst case, where N_A is the number of attributes in the generalized relation R' and N' is the number of tuples in R' . Usually N' is not large in the generalized relation R'.

3 Attribute-Oriented Induction in Databases

In this section, we discuss how to generalize the primitive data in database system by attribute-oriented induction. The general idea of basic attribute-oriented induction is one in which generalization is performed attribute by attribute using attribute removal and concept tree ascensioni[1]. As a result, different tuples may be generalized to identical ones, and the number of distinct tuples in the generalized relation is reduced. In the procedure of generalization, the tuples in database are generalized to the desirable level, the table gained at this stage is called generalized relation.

Definition 3.1 An attribute is generalizable if there are a large number of distinct values in the relation but there exists a concept hierarchy for the attribute (i.e., there are higher level concepts which subsume these attribute values). Otherwise, it is nongeneralizable.

Definition 3.2 An attribute in a relatively large relation is desirable for consideration for generalization if the number of distinct values it contains does not exceed a user-specified desirability threshold (usually ≤ 6).

If an attribute is nongeneralizable, then it should be removed in the generalization. Attribute removal corresponds to the generalization rule, *dropping conditions* [7]. Consider a tuple as a set of conjuncts in the logical forms; an attribute value together with its attribute name form one of the conjuncts. By removing a conjunct, a constraint is eliminated and the concept is generalized. If there are a large set of distinct values for an attribute, the large set of values must be generalized. However, if there is no higher level concept provided for the attribute, it can not be further generalized by ascending the concept tree. Therefore, the attribute should be eliminated in generalization. Attribute removal can also be viewed as a generalization of the attribute to the most general concept ANY and then removed from the representation.

If an attribute is generalizable, then it should be generalized to a higher level concept value by concept tree ascension techniques. Concept tree ascension corresponds to the generalization rule, *climbing generalization trees* [7]. If there exists a higher level concept for the value in the concept tree, then the substitution of the value in each tuple in the relation by the corresponding higher level concept makes the tuple cover more cases than the original one, and thus it generalizes the tuple.

A *generalized relation* R_g for a set of data R stored in the relational table is an intermediate relation generalized from the relation R by removing nongeneralizable attributes and generalizing each generalizable attribute to a *desirable level.* Let a *desirability threshold* be available for each attribute, which could be set by default or specified by the user or an expert, based on the semantics of the attributes and/or the expected forms of generalized rules (usually ≤ 8).

Algorithm 2 (Generalization Algorithm:) *Extraction of the generalized relation from a set of data R*

Input: (i) A set of task-relevant data R (assume that they are obtained by a relation query and are stored in a relation table), a relation of arity n with a set of attributes A_i ($1 \leq i \leq n$); (ii) a set of concept hierarchies, H_i, where H_i is a hierarchy on the generalized attribute A_i, if available; and (iii) a set of desirability thresholds T_i for each attribute A_i

Output. The generalized relation R_g

Method

```
1. $R_t := R$; /* $R_t$ is a temporary relation. */
   /* Suppose $d_i$ is the number of distinct values in attributes $A_i$ */
   begin
2. for each attribute $A_i$ ( $1 \leq i \leq n$) of $R$ do {
      while $T_i \leq d_i$ do
      if there is no higher level concepts of $A_i$ then remove $A_i$;
      else substitute the values by its corresponding minimal
            generalized concept; }
      eliminate redundant tuples
```

/* Generalization is implemented as follows. First, collect the distinct values in the relation and compute the lowest level L on which the number of distinct values will be no more than T_i by synchronously ascending the concept hierarchy from these values. Then generalize the attribute to this level L by substituting for each value A_i's with its corresponding concept H_i at level L. */ }

3. $R_g := R_t$

4 DBROUGH System

DBROUGH is a descendant of DBLEARN [1,6]. The architecture of the system is shown in Figure 1. The system can discover different rules from relational databases, including characteristic rules, discrimination rules and decision rules. Also it can find all the maximal generalized rules in the data. The system takes SQL-like database learning requests and perform different algorithms to find different rules. The background knowledge is stored in a concept hierarchy table. The provided concept hierarchies can be adjusted dynamically according to database statistic and specific learning request.

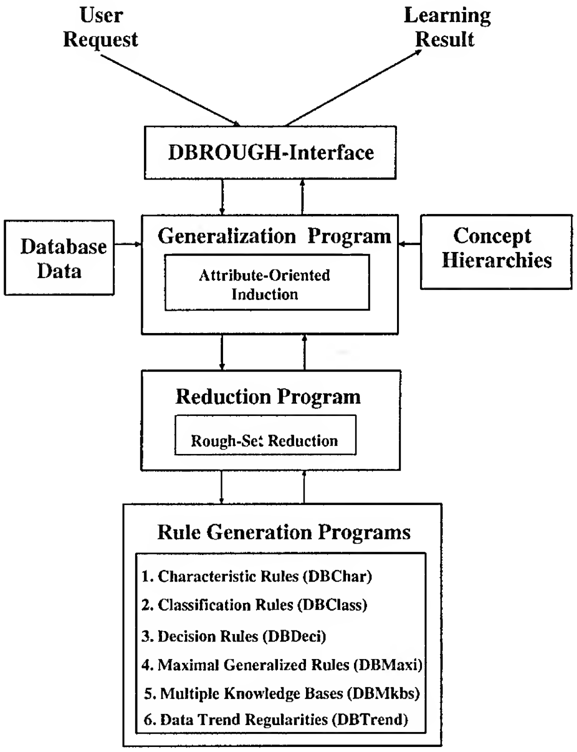

User Request
Learning Result
DBROUGH-Interface
Database Data
Generalization Program
Attribute-Oriented Induction
Concept Hierarchies
Reduction Program
Rough-Set Reduction
Rule Generation Programs
1. Characteristic Rules (DBChar)
2. Classification Rules (DBClass)
3. Decision Rules (DBDeci)
4. Maximal Generalized Rules (DBMaxi)
5. Multiple Knowledge Bases (DBMkbs)
6. Data Trend Regularities (DBTrend)

Figure 1: The Architecture of DBROUGH

In order to constrain a knowledge discovery process to generalization on a particular set of data using a particular set of background knowledge, learning should be directed by specific requests. A database learning request should consist of (i) a database query which extract the relevant set of data, (ii) the kind of rules to be learned, (iii) the specification of the target class and possibly the contrasting classes depending on the rules to be learned, (iv) the preferred concept hierarchies, and (v) the preferred form to express learning results. Notice that (iv) and (v) are optional since default concept hierarchies and generalization threshold values can be used if no preferred is specified explicitly.

In our system DBROUGH, the learning procedure is initiated by a user learning request. The learning request can be viewed as an extension to relational language SQL for knowledge discovery in databases. In this paper we only discuss how to combine rough set theory to discover decision rules (**DBDeci**) in section 5 and maximal generalized rules (**DBMaxi**). in section 6. The interested reader should refer to [1,6] to get the detailed information about how to discover characteristic rules and discrimination rules.

5 Learning Decision Rules

In this section, an algorithm **DBDeci** for discovering decision rules using rough set theory is introduced. Suppose our objective is to learn a decision rule which tell which features of a car really determine the mileage. The request is specified to DBROUGH as follows:

learn decision rule
for *Mileage*
from *Car_relation*

Notice in this learning request, the concept hierarchies and threshold are not specified, thus the default ones will be used.

DBROUGH transfers the user's learning request to SQL, then extracts the data from the relation (**Car_relation**) and the result is obtained. Then apply generalization algorithm, we get the generalized table.

After the generalization process, rough set method is performed on the generalized relation table. First the core of the attributes is computed, then the best reduct or the user minimal attribute subset of the attributes can be constructed by applying reduct algorithm. The reduction of the generalized relation is performed further by removing those attributes which are not in the reduct or the user minimal attributes subset and thus simplify the generalized relation.

Strategy 1: *find the desired reduct or user minimal attributes and reduce the generalized relation.*

Strategy 2: *combine the similar tuples.*

Strategy 3: *Transform the tuples in the reduced relation into decision rules for each class.*

Algorithm 3 *DBDeci-An Attribute-Oriented Rough Set Approach for Learning Decision Rules in Databases*

Input: (i) A set of task-relevant data R (assume that they are obtained by a relation query and are stored in a relation table), a relation of arity n with a set of attributes $C = \{c_i\}$ ($1 \leq i \leq n-1$) and decision attribute D (ii) a set of concept hierarchies, H_i, where H_i is a hierarchy on the attribute c_i, if available; (iii) the class threshold value T

Output. A set of decision rules for each class of D

Method

Step 1. *Attribute-oriented induction.* (generalization algorithm)

Step 2. *find the best reduct or user minimal attribute subset with respect to D* (Reduct Algorithm)

Step 3. *Reduce the generalized relation by removing those attributes which are not in the reduct or user minimal attributes subset*

Step 4. *Combine the similar tuples in the reduced relation*

Step 5. *Transform the tuples in the reduced relation into decision rules for each class in D*

6 Computing Maximal generalized Rules

The maximal generalized rules minimize the number of rule conditions and are in a sense optimal because their conditions are non-redundant. We use RUL to denote the collection of all maximal generalized rules for the decision V_d.

The algorithm **DBMaxi** compute the maximal generalized rules as follow: for a real large database, first, the generalization algorithm is applied. After the generalization process, rough set method is performed on the generalized relation. The decision matrix[12] for the decision values of the decision attribute are constructed and the maximal generalized rules are computed from them.

6.1 Decision Matrix

For the selected decision attribute $d \in A$ and its particular value V_d, we will focus on the collection of objects e (the concept), for which $d(e) = V_d$, i.e., the set $|V_d|$. Before attempting to find discriminating rules for $|V_d|$ in terms of other attributes belonging to $A - \{d\}$, we will summarize all the attribute-value pairs distinguishing objects belonging to $|V_d|$ and $U - |V_d|$ in the matrix format defined as follows.

Definition 6.1 let e_i denotes any object belonging to $|V_d|$, i.e., $i = 1, 2, ...,$ $Card(|V_d|) = \rho$ and let $e_j \in U - |V_d|$, $j = 1, 2, ..., card(U - |V_d|) = \gamma$. the decision matrix $DM = (DM_{ij})_{\rho\times\gamma}$ is defined as $DM_{i,j} = \{(a, a(e_i)) : a(e_i) \neq a(e_j)\}$.

The set $DM_{i,j}$ contains all pairs whose values are not identical on both e_i and e_j. In other words, $DM_{i,j}$ represents the complete information distinguish e_i and e_j. The distinguishing attributes for different combination of i and j can be represented in the form of a matrix $DM = [DM_{ij}]_{\rho\times\gamma}$.

6.2 Decision Matrix and Maximal Generalized Rules

Let $e_i \in |V_d|$, we will use the symbol RUL_i to denote the set of all maximal generalized rules whose conditions match the features of object e_i, that is

$$RUL_i = \{r \in RUL : A_r(e_i) = V_r\}$$

Clearly, if the collection of rules RUL_i is known for each $e_i \in |V_d|$ then all the maximal generalized rules for target decision $|V_d|$ can be obtained by taking the union $RUL = \bigcup_i RUL_i$.

The maximal generalized rules can be computed by simplifying an associated Boolean function called decision function[12]. The decision function B_i is constructed out of the row i of the decision matrix, that is, $(DM_{i1}, DM_{i2}, ..., DM_{ir})$ by formally treating each attribute-value pair occuring in component DM_{ij} as a Boolean variable and then forming Boolean conjunction of disjunctions of components belonging to each set DM_{ij} $(j = 1, 2, ..., \gamma)$. That is,

$$B_i = \bigcap_j \bigcup DM_{ij},$$

where $\bigcap$ and $\bigcup$ are respectively generalized conjunction and disjunction operators.

By applying the distributivity and absorption laws of Boolean algebra, each decision function can be expressed in a simplified form of a disjunction of minimal conjunctive expressions. We can derive the general procedure for computing all maximal generalized rules for the given target decision[12]. The procedure requires the construction of the decision matrix for each target decision prior to computation of rules. Once the decision matrix have been constructed the key steps to compute the rules are summarized in algorithm 4 (DBMaxi).

Algorithm 4 *DBMaxi: Compute the maximal generalized rules*

Input: a relational system R
Output. all the maximal generalized rules
Method
Step 1: Extract the generalized relation R' from R (Generalization Algorithm)
Step 2: Compute decision matrix for the current decision category in R'
Step 3: For each positive case $e_i, (i = 1, 2, ..., \rho)$ compute the set of all maximal generalized rules MIN_i matching this case by evaluating and simplifying (using the absorption law) the associated decision function B_i.
Step 4: Compute the union $\cup MIN_i$ of maximal generalized rule sets to find all maximal generalized rules for the current decision category.

7 Conclusion

In this paper, we further develop our previous research work in knowledge discovery in database by using rough set theory [8]. We discussed two algorithms to compute the decision rules (DBDeci) and maximal generalized rules (DBMaxi). Our method use attribute-oriented induction for generalization, which provides an efficient way to generalize the database and greatly reduce the computational complexity, the time complexity of our method is O(NlogN), where N is the number of the tuples. Most learning algorithms in the literature [3] are tuple-oriented algorithms. A tuple-oriented method examines data in the database tuple by tuple and performs generalization based on the comparison of tuple values with the intermediate generalization results. Since the number of the possible tuple combinations is exponential to the number of tuples in the relevant data set, the worst case complexity of the generalization process is exponential to the size of the relevant data sets.

Moreover, our method integrates the generalization and reduction process. Our algorithm eliminates the undesirable attributes in the generalization process and get rid of the irrelevant or superfluous attributes in the reduction process, the rules generated are very concise, expressive and strong. Besides, with concept generalization, the derived rules are represented in higher level abstraction.

References

1. Y. Cai, N. Cercone and J. Han, Attribute_Oriented Induction in Relational databases, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds) pp. 213-228, 1991.
2. T.G. Dietterich and R.S. Michalski, A Theory and Methodology of Inductive Learning, *in Machine Learning: An Artificial Intelligence Approach, Vol. 1.* Michalski et. al. (eds), Morgan Kaufmann, pp 43-82, 1983.
3. W. J. Frawley, G. Piatetsky and C.J. Matheus, Knowledge Discovery in Database : An Overview, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds), pp. 1-27, 1991.
4. Wojciech Ziarko, The Discovery, Analysis, and Representation of Data Dependencies in Databases, in *Knowledge Discovery in Databases* G. Piatetsky-Shapiro and W. J. Frawlwy,(eds) Menlo Park, CA: AAAI/MIT, 1990, 213-228
5. J. Han, Y.Cai, N. Cercone, Knowledge Discovery in Databases: An Attribute-Oriented Approach, *Proceeding of the 18th VLDB Conference*, Vancouver , B.C., Canada, pp 340-355, 1992
6. X. Hu, N. Cercone, J. Han, A Rough Set Approach for Knowledge Discovery in Databases, *The International Workshop on Rough Set and Knowledge Discovery*, Banff, Alberta, Canada, October 12-15, 1993. pp79-94
7. R.S. Mickalski, A Theory and Methodology of Inductive Learning, *in Machine Learning: An Artificial Intelligence Approach, Vol. 1.* Michalski et. al. (eds), Morgan Kaufmann, 1983, pp 41-82.
8. Zdzislaw Pawlak, Rough sets, *International Journal of Information and Computer Science* (1982) 11(5), 341-356
9. Zdzislaw Pawlak, Rough Classification, *International Journal of Man-Machine Studies* (1984) 20, 469-483
10. A. Silberschatz, M.Stonebraker and J.D.Ullman, Database Systems:Achievements and Opportunities, *Comm. ACM, 34(10), 1991*, pp. 94-109.
11. A. Skowron, C. Rauszer, The discernibility matrices and functions in information systems. *ICS Research Report 1/91*, Wawsaw University of Technology, Nowowiejska 15/19, 00-665, Warsaw, Poland
12. Wojciech Ziarko, Ning Shan, A Rough Set-Based Method for Computing All Minimal Determintic Rules on Attribute-Value Systems, *Technical Report CS-93-02* Dept. of Computer Science, University of Regina, Canada

Restructuring rule bases to improve performance

Alex Lopez-Suarez and M. Kamel
Pattern Analysis and Machine Intelligence (PAMI) Lab
Department of Systems Design Engineering,
University of Waterloo,
Waterloo, Ont., Canada. N2L 3G1.
alopez@watnow.uwaterloo.ca mkamel@watnow.uwaterloo.ca

ABSTRACT

This paper presents a new methodology to restructure rule bases through the combination of Explanation-Based Learning (EBL) and knowledge abstraction techniques. Performance improvements resulting from restructuring are assessed in terms of pattern matching activity during problem solving. The introduction of redundancy that results as a side effect of the restructuring techniques is discussed and algorithms are presented to control it. Examples and experimental results using typical problems are presented.

Keywords
Adaptive Systems, Learning, Knowledge restructuring, Inferential reasoning

1. Introduction

Expert systems utilize knowledge about specific domains and inferential reasoning to solve problems. Knowledge is kept separate from the mechanism which manipulates the knowledge to solve problems (the inference engine). Knowledge is commonly represented as a collection of rules and related facts. Rules are activated based on facts initially available. Inferences made by the rules generate new facts that cause the activation of other rules. This inferential approach to problem solving is intuitively appealing. Furthermore the structure of individual rules is conceptually simple. The left-hand side represents conditions that must be present for the rule to be activated, and elements on the right-hand side are actions performed when the rule is executed. However, the behavior of large rule-based systems is hard to predict because interactions which occur between the rules are not obvious. Measuring the impact of organizing the rules in the knowledge base in different ways is also difficult to assess.

Knowledge bases can be organized for several purposes. Other researchers have proposed restructuring methods to allow the verification and validation of rule bases [3,7], to facilitate their maintenance [8], or to explain their actions [15,16]. The methodology proposed in this paper restructures rule bases with the objective of improving the performance of expert systems during problem solving. It consists of combining explanation-based learning and knowledge abstraction techniques. An alternative strategy for combining these techniques in the context of the PRODIGY system is presented in [10]

Explanation-based learning (EBL) identifies a wide variety of deductive learning methods. The EBL methodology evolved from research in concept definition, problem solving, planning, and learning [1,11,12,13]. Typically EBL algorithms analyze positive examples in the context of existing knowledge in a given domain. The EBL methodology consists of two stages. First a deductive explanation of the example is generated identifying elements of the knowledge base required to justify the solution provided. In the second stage the explanation is analyzed to construct a more efficient solution.

Abstraction is commonly understood as a mechanism for identifying important aspects of a problem and ignoring details. Research in planning and learning utilize abstraction to develop problem solving hierachies to reduce the search for solutions [9,14,17]. Typically abstraction is applied in goal-oriented, backward chaining, inference systems. Abstraction techniques usually consist of removing or weakening the applicability of rule conditions. In this paper abstraction is utilized to increase the aplicability of EBL rules by removing common conditions.

The methodology presented in this paper utilizes EBL to compress a sequence of rules that solves a specific problem into a single rule. Since each EBL rule is usually applicable only to a small set of problems, we use abstraction to combine EBL rules creating new rules that are applicable to larger problem subsets. EBL and abstraction are used as complementary operations for finding an optimal structure of a knowledge base in terms of the number of rules present. Assessment of performance improvement is made by analyzing the effect of changes in pattern matching activity and number of rules executed during the operation of the inference engine. The methodology is designed to work on forward chaining rule based systems that use the Rete Pattern Matching algorithm [5] to implement the inference engine.

The paper is organized as follows. Section 2 presents the methodology. Section 3 describes the evaluation of performance improvements. Section 4 analyzes mechanisms to control or avoid redundancy. Section 5 describes the implementation and experiments conducted, and section 6 discusses results.

2. Restructuring Rule Bases using EBL and Abstraction

The structure of a rule base is determined by the elements of each rule and the interactions that occur between the rules during problem solving. A given rule base structure can be modified by subdividing rules into smaller rules, grouping them to create composite rules, changing the number of elements in rules, or even simply changing the order of elements within a rule. Any of these structural changes affect the performance of the expert system. The approach proposed in this paper is to combine rule composition and decomposition to adapt the rule base to classes of problems. The restructuring process is guided by the analysis of how the expert system utilizes knowledge to solve problems. The methodology relies on two sources of information, the contents of the knowledge base and traces of the actions performed by the expert system to solve problems.

A rule-based expert system E is defined by a triplet (K, I, W) where K represents the domain knowledge as a set of rules, I is the inference engine used to apply rules in K to solve problems, and W is the working memory where facts (representing available data) are stored. Each rule is a condition-action pair (C,A) of the form C *implies* A. The condition set C must be satisfied before the action set A can be executed. The facts stored in W are predicates represented by a predicate name and a list of arguments. In forward chaining systems the inference engine I identifies and executes rules based on facts initially available applying the cycle (match facts-conditions, select a rule, execute rule, update W). For our purposes a solution to a problem is defined as a triplet (D, T, G) where D represents initial data available in W, T is the solution trace, and G represents the expected final contents of W, including the solution to the problem (the goal). The solution trace consists of a sequence of states $T=\{(W_i,C_i,A_i), i=1,n\}$ where W_i represents the facts available at state i, and (C_i,A_i) the condition and action sets of the rule selected for execution.

The method analyzes problem solution sequences in two stages: Rule compression, and rule abstraction. Processing prior to these stages removes rule executions that led to incorrect paths during the search for a solution and, consequently, had to be backtracked. This preliminary process is applicable only to cases where rules interact recursively during the search for solutions and is not discussed further in this paper.

Stage 1 compresses a problem solution sequence $S = (D=W_1,\ T=\{(W_i,C_i,A_i),\ i=1,n\},\ G=W_n)$ into a single rule r_e using EBL. Initially $r_e = (C_n,A_n)$. The remaining rules are unified in reverse order of execution, progressively updating the contents of r_e. At each iteration of the algorithm three elements (W_{i-1},A_{i-1},C_i) are combined. Assertions made by intermediate rules are added to the RHS of the new rule. The algorithm transforms S into $(D=W_1,r_e,G=W_n)$, providing a direct path to solve a problem that originally required the application of several rules. The effect of adding r_e to K in terms of performance improvement and rule redundancy are analyzed using the techniques described in sections 3 and 4. Problem solution sequences used to derive EBL rules are kept to allow rule base restructuring using abstraction. Redundant rules are kept separate from the active rule base.

Stage 2 utilizes abstraction to combine pairs of EBL rules that reach the same goal. Given two EBL rule derivations $(D_1,r_{e1}=(C_1,A_1),G)$ and $(D_2,r_{e2}=(C_2,A_2),G)$, the abstraction algorithm generates three rules $r_1=(C_1\text{-}C_2,A_{r1})$, $r_2=(C_2\text{-}C_1,A_{r2})$, $r_3=(C_1\cap C_2 \cup (A_{r1} \mid A_{r2}),\ A_{r3})$. Abstraction can be applied iteratively to EBL rules and to original rules with common goals. If $C_1\cap C_2 = \emptyset$ abstraction is not applicable. Abstraction generates a single rule if one of the input rules is subsumed by the other. Similar to stage 1, efficiency and redundancy analyzes are performed on the restructured knowledge base after abstraction is completed.

Domain Theory	**Properties of object cp1**	**Properties of object cp2**
r1-cup	*(small cup1)*	*(small cup2)*
(hold-liquid object)	*(sides cup1 s1)*	*(sides cup2 s2)*
(drink-from object)	*(made-from s1 plastic)*	*(made-from s2 china)*
(stable object)	*(bottom cup1 b1)*	*(bottom cup2 b2)*
(liftable object)	*(made-from b1 styrofoam)*	*(made-from b2 metal)*
=> assert (cup object)	*(flat b1)*	*(flat b2)*
r2-hold-liquid	*(has cup1 c1)*	*(has cup2 c2)*
(sides object s)	*(concavity c1)*	*(concavity c2)*
(made-from s ms)	*(upward-pointing c1)*	*(upward-pointing c2)*
(non-porous ms)	*(cylindrical s1)*	*(has cup2 h2)*
(bottom object b)	*(non-porous plastic)*	*(handle h2)*
(made-from b mb)	*(non-porous styrofoam)*	*(non-porous china)*
(non-porous mb)	*(light-material plastic)*	*(non-porous metal)*
=> assert (hold-liquid object)	*(light-material styrofoam)*	*(light-material china)*
r3-can-drink-from	*(insulating-material plastic)*	*(light-material metal)*
(has object obj1)	*(insulating-material styrofoam)*	
(concavity obj1)	**Rules executed**	**Rules executed**
(upward-pointing obj1)	*r7-is-insulated*	*r6-is-lightweight*
=> assert (drink-from object)	*r6-is-lightweight*	*r2-hold-liquid*
r4-is-stable	*r5-is-liftable*	*r8-has-handle*
(bottom object b)	*r2-hold-liquid*	*r5-is-liftable*
(flat b)	*r3-can-drink-from*	*r3-can-drink-from*
=> (assert (stable object)	*r4-is-stable*	*r4-is-stable*
r5-is-liftable	*r1-cup*	*r1-cup*
(light-weight object)	**EBL rule**	**EBL rule**
(graspable object)	*(small v707)*	*(small v707)*
=> assert (liftable object)	*(sides v707 v106)*	*(sides v707 v106)*
r6-is-lightweight	*(cylindrical v106)*	*(made-from v106 v110)*
(small object)	*(made-from v106 v113)*	*(light-material v110)*
(sides object s)	*(insulating-material v113)*	*(bottom v707 v117)*
(made-from s ms)	*(light-material v113)*	*(made-from v117 v121)*
(light-material ms)	*(bottom v707 v217)*	*(light-material v121)*
(bottom object b)	*(made-from v217 v221)*	*(non-porous v110)*
(made-from b mb)	*(light-material v221)*	*(non-porous v121)*
(light-material mb)	*(non-porous v113)*	*(has v707 v306)*
=> assert (light-weight object)	*(non-porous v221)*	*(handle v306)*
r7-is-insulated	*(has v707 v503)*	*(has v707 v503)*
(small object)	*(concavity v503)*	*(concavity v503)*
(sides object s)	*(upward-pointing v503)*	*(upward-pointing v503)*
(cylindrical s)	*(flat v217)*	*(flat v117)*
(made-from s m)	*=>*	*=>*
(insulating-material m)	*assert (graspable v707)*	*assert (light-weight v707)*
=> assert (graspable object)	*assert (light-weight v707)*	*assert (hold-liquid v707)*
r8-has-handle	*assert (liftable v707)*	*assert (graspable v707)*
(small object)	*assert (hold-liquid v707)*	*assert (liftable v707)*
(has object b2)	*assert (drink-from v707)*	*assert (drink-from v707)*
(handle b2)	*assert (stable v707)*	*assert (stable v707)*
=> assert (graspable object)	*assert (cup v707)*	*assert (cup v707)*

Figure 1. A simple domain theory and generation of two EBL rules.

Example

In order to illustrate the application of the methodology, consider a simple domain theory defining the concept of a cup shown in Figure 1. The condition sets of the rules specify properties that an object must have to be recognized as a cup. The known properties of two objects, the rule execution sequences leading to their recognition as valid cups, and the resulting EBL rules are also included in the Figure.

The recognition process starts by matching available facts with conditions in rules leading to the assertion of additional facts. Through the execution of rules the objects *cp1* and *cp2* are recognized as valid cups according to the domain theory.

Stage 1 applies the EBL algorithm to reduce an execution sequence to a single rule. The condition patterns of the EBL rule specify essential properties from the initial data and include intermediate assertions made through the rule executions. The algorithm keeps the condition patterns as general as possible (by introducing variable names such as *v707*) to allow the recognition of a type of cup, rather than the single description given by the example data. *cp1* identifies insulated cups, and *cp2* cups that have handles.

Stage 2 applies the abstraction algorithm to identify features shared by the two EBL rules constructing the abstract rule shown below. Condition patterns that differ are used to identify rules *r7-is-insulated* and *r8-has-handle* in the solution sequences that have these patterns. These rules are reintroduced in the knowledge base. The abstract rule has the patterns that are common in *cp1* and *cp2* and the pattern *(graspable v707)* asserted by the rules added. This pattern is removed from the right-hand side of the abstract rule.

Abstract Rule cp12

```
(small v707 )                  =>  assert (light-weight v707 )
(sides v707 v106 )                 assert (liftable v707 )
(made-from v106 v113 )             assert (hold-liquid v707 )
(light-material v113 )             assert (drink-from v707 )
(bottom v707 v217)                 assert (stable v707 )
(made-from v217 v221 )             assert (cup v707 )
(light-material v221 )
(non-porous v113 )
(non-porous v221 )
(has v707 v503 )
(concavity v503 )
(upward-pointing v503 )
(flat v217 )
(graspable v707 )
```

In summary, the methodology utilizes compression to create new rules that establish direct paths from initial data to solutions. Abstraction creates hierarchies that group rules by common data requirements. Abstraction and compression are be viewed as complementary operations. Rules removed from the active rule base are kept to allow further restructuring.

3. Performance Evaluation of Rule base Restructuring

Evaluation of efficiency resulting from the application of EBL and abstraction is determined by analyzing changes in performance. This is done by evaluating pattern matching activity in the pattern and join networks created by the rete algorithm [5]. The evaluation method measures pattern-fact matches and partial pattern matches. Node sharing within the pattern and join networks is taken into account as well as the computation of multiple matchings due to the presence of duplicate facts. Rule activations due to the addition of new rules is also considered. Knowledge base improvements are determined by measuring the performance of the expert system before and after structural changes are introduced.

3.1 Efficiency of Rule Compression

To evaluate performance changes attained from rule restructuring using rule compression we apply formula 1. This formula computes the utility of restructuring as a percentage of the relative costs of solving a problem with two knowledge base structures.

$$U = 100\,(C_K - C_R) / C_k \qquad \textbf{(1)}$$

where,

$$C_K = M_K + P_K - P_J$$

$$C_R = M_R + P_R$$

C_K is the cost of solving the problem with the original knowledge base K and C_R the cost of solving it after the compressed rule r_e is added to K. M_K and M_R refer to the number of pattern-fact matches performed by the solution sequence and the compressed rule respectively. Similarly, P_K and P_R account for the number of partial pattern matches. P_J is the cost of partial matches in join nodes shared by the condition sets of rules in the solution sequence. This cost is subtracted because the computation of P_K counts matches in join nodes for each rule, thus counting repeatedly shared joins. The formula does not include partial matches in join nodes shared between patterns in the compressed rule and rules activated by its presence since they rarely occurred in experiments conducted.

3.2 Efficiency of Abstraction

The efficiency of an abstract rule is measured in relation to the input rules from which it is derived. Performance changes due to abstraction are evaluated as follows. We first restructure the rule base by introducing the abstract rule and any other rules identified during its creation. Next the expert system is executed with the data sets used by the two rules from which the abstract rule was derived. Each execution trace is used to create an EBL rule and its performance efficiency is computed using Formula 1. This process effectively reverses abstraction reproducing the initial input rules. Each utility value computed represents the change in performance resulting from using of the abstract rule instead of one of the initial rules.

Typically abstraction creates rules that are more general than its input rules. The rules are applicable in solving more problems. However they are not as efficient as EBL rules, since additional rules need to be executed to complete problem solutions. Thus, there is a tradeoff between restructuring a rule base so that it has few general rules versus one that has many rules each solving a single problem or only a small subset of problems.

4. Controlling Redundancy

Restructuring rule bases using EBL and abstraction techniques introduces redundancy. EBL uses a collection of selected rules to create a new rule. Abstraction combines EBL rules to create more general rules. Redundant rules cause problems in other activities, such as knowledge base maintenance, verification, and reliability. Therefore it is important to identify rules that become unnecessary after restructuring aimed at improving efficiency is completed.

Removal of redundancy is straightforward when the abstraction technique is used. The input rules used by the algorithm are effectively replaced by the abstract rule and the additional rules introduced. Therefore the input rules become redundant and can be removed from the active knowledge base.

Detecting redundancy when EBL rules are introduced is more difficult. In general given a solution sequence $S = (D=W_1,\ T=\{(W_i,C_i,A_i),\ i=1,n\},\ G= W_n)$ that generates a compressed rule r_e, it is not possible to identify redundant rules unless some information about the uses of the knowledge base is available. Consider the removal of the last rule executed. Removal of this rule would prevent the system from reaching the goal when the assertions made by the previous rule are made available as data. Attempting to remove any other rule in the solution sequence would prevent the activation of other rules in the knowledge base that depend on the assertions that it makes.

To develop algorithms for detecting redundant rules when EBL rules are added to the rule base, the following two assumptions need to be established:

A1 *Intermediate inferential conclusions cannot be provided directly as data*
A2 *Only final results are needed*

If these two assumptions hold, rules that are potentially redundant are those whose RHS assertion patterns are only needed by rules in the solution sequence. If just the first assumption can be made then only the last rule in the solution sequence is eligible for removal. This rule becomes redundant if all its condition patterns are matched by facts asserted through a single rule, since this rule must be part of the solution sequence. When a problem domain has the second assumption alone, only rules whose condition patterns are matched directly by data can be removed.

The algorithms require as input parameters the knowledge base K and the sequence used to generate the EBL rule r_e. To allow for different knowledge base characteristics, three algorithms are developed, one for each assumption and a third combining both assumptions. To allow an efficient detection of redundancy, initially two pattern lists are created, one for assertions and one for conditions in rules.

5 Implementation and Experiments

A software system prototype, called DyKOr (Dynamic Knowledge Reorganization), has been developed to study the effects of restructuring rule bases using the methodology presented. This software accepts rule bases and solution sequences generated using CLIPS [6] as input. CLIPS is a typical forward chaining rule based system that employs the Rete algorithm to implement its inference engine. DyKOr is implemented in standard C language and is separate from CLIPS. The software consists of four separate modules that implement (i) the EBL algorithm, (ii) the abstraction algorithm, (iii) the evaluation of efficiency, and (iv) the algorithms to control redundancy. The software executes interactively to allow experimentation with alternative rule base structures.

The implementation of EBL and abstraction algorithms includes processing of logical connectives, comparisons tests, and manipulation of variables The module to evaluate efficiency considers only the part of pattern and join networks storing rules used to generate the EBL rule. Thus avoiding analysis of the complete rule base. The system keeps solution sequences and rules removed separate from the active knowledge base.

Problem	**M_K**	**P_K**	**P_J**	**M_R**	**P_R**	**Efficiency Gain**
cp1	26	22	1	20	14	27.66%
cp2	25	21	0	19	16	23.91%
cp3	25	22	1	19	12	32.61%
Abstract(cp1,cp2)	16	17	1	15	14	9.38%
Abstract(cp1,cp3)	20	14	0	20	14	0%
CarDiagnosis	12	21	15	6	5	33.33%
BlocksWorld	20	14	0	10	9	44.12%
SafeToStack	6	5	0	3	2	54.55%

Figure 2. Efficiency of EBL and abstract rules in Rete networks.

The table shown in Figure 2 summarizes the results of several experiments. The first five rows refer to the rule base described in section 2.1. The execution sequence that generates rule *cp1* requires a total of 26 pattern-fact matches and 22 partial pattern-fact matches. One of the join nodes created by these rules is shared. The EBL rule requires 20 pattern-fact matches and 14 partial pattern-fact matches. Applying formula 1 a performance efficiency gain of 27.66% is attained when the EBL rule is introduced to the rule base. To compute the efficiency of the abstract rule *cp12*, derived from rules *cp1* and *cp2*, the system is run with the abstract rule included and the data used to derive the rule *cp1*. The efficiency gain is 9.38%. Thus, solving the problem with the abstract rule is less efficient than solving it with the EBL rule. The efficiency of the abstract rule from (*cp1,cp3*) is zero because *cp3* is a subset of *cp1*. (rule *cp1* specifies cups made of two different insulating materials, and *cp3* defines cups without handles made a single insulating material).

Other experiments performed include *CarDiagnosis*, a typical rule base commonly found in expert system literature. The *SafetoStack* and *BlocksWorld* problems are drawn from research in planning and EBL [12,13,4]. Cases where EBL produces a decrease in performance efficiency are not included. Experimental results presented are consistent with results obtained by other researchers [2,10].

6 Discussion

The methodology presented proposes the use of deductive learning to adapt the behavior of expert systems based on their actual use. The approach followed relies on the existence of a domain theory which is used to justify and then generalize the solution to a training example. This approach is in contrast with classical learning techniques based on similarities across a set of training examples. Explanation-based learning allows the removal of irrelevant data and the construction of composite rules to reach specific goals. The combination of EBL with abstraction allows the restructuring of the rule base into hierarchies of rules applicable to classes of problems. Modifications made to the rule base through the methodology are guaranteed to be consistent with the domain theory.

Limitations of the methodology derive from the fundamental assumptions made by the techniques employed. EBL requires positive examples and a complete and correct domain theory. The effect of these limitations is discussed elsewhere [1,13]. Abstraction assumes the existence of rule subsets leading to common goals. This limits the extent to which the method can be used to construct rule hierarchies. The algorithms to detect redundancy require information about the uses of the knowledge base which may not be readily available or may change over time. Furthermore, the methodology requires the maintenance of a separate knowledge base to store inactive rules.

The Analysis of performance improvements evaluates only changes in pattern matching activity. Experiments have led to the identification of other factors that affect performance, including characteristics of initial data and the number of rules executed to reach a solution. These effects are currently being investigated.

References

[1] DeJong, G. F., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1(2), 145-176.

[2] Doorenbos, B., Tambe, M., & Newell, A. (1992). Learning 10,000 chunks: what's it like out there? *Proceedings of the Tenth National Conference on Artificial Intelligence*, 830-836.

[3] Evertsz, R. (1991). The automated analysis of rule based systems, based on their procedural semantics. In *Proceedings IJCAI-91*, (pp. 22-27).

[4] Flann, N. S., & Dietterich, T. D. (1989). A study of explanation-based methods for inductive learning. *Machine Learning*, 4, 187-226.

[5] Forgy, C. (1982). A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19, 17-37.

[6] Giarratano, J. C. (1991). CLIPS User's Guide Version 5.1. NASA-Johnson Space Center, Houston, Tx.

[7] Ginsberg, A. (1988). Knowledge base reduction: A new approach to checking knowledge bases for inconsistency and redundancy. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, (pp. 585-589).

[8] Jacob, R. J. K., & Froschei, J. N. (1990). A software engineering methodology for rule-based systems. *IEEE Trans. on Knowledge and Data Engineering, 2(2), 173-189.*

[9] Knoblock, C. A. (1991) *Automatically generating abstractions for problem solving*. PhD, Carnegie Mellon University.

[10] Knoblock, C. A., Minton, S., & Etzioni, O. (1991). Integrating abstraction and explanation-based learning in PRODIGY. *Proceedings of the Ninth National Conference on Artificial Intelligence*, 541-546.

[11] Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking is Soar: The anatomy of a general learning mechanism. *Machine Learning*, 1(1), 11-46.

[12] Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42, 363-391.

[13] Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-Based Generalization: A Unifying view. *Machine Learning*, 1(1), 47-80.

[14] Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 55(2), 115-135.

[15] Swartout, W., Paris, C., & Moore, J. (1991). Design of explainable expert systems. _*IEEE Expert*, 58-64.

[16] Tanner, M. C., & Keuneke, A. M. (1991). Explanations in knowledge systems: The roles of task structure and domain functional models. *IEEE Expert*, 50-57.

[17] Unruh, A., & Rosenbloom, P. S. (1989). Abstraction in problem solving and learning. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 681-687.

Learning Heuristics for Ordering Plan Goals through Static Operator Analysis

T. L. McCluskey and J. M. Porteous
lee@zeus.hud.ac.uk, julie@zeus.hud.ac.uk

The School of Computing and Mathematics,
The University of Huddersfield,
Queensgate, Huddersfield HD1 3DH, UK

Abstract. There is a trade-off between the generality and efficiency of automatic planning systems which means that general planners tend to be inefficient, the problem of "efficiency versus generality". Here we present PRECEDE, a novel method of off-line compilation, that analyses domain operators and produces heuristics that reduce search during plan generation. We present a declarative specification of PRECEDE, illustrate it with a worked example and discuss the results of some empirical tests and related work.

1 Introduction

The design and implementation of automatic planning systems has been the focus of much work over the past 25 years or so and during that time the term has come to cover a wide area of research activity. In this paper, we will concentrate on *classical* planning systems in the tradition of STRIPS [6] which are *generative*: they generate entire plans before any actions in them are executed and *domain independent*: there is a logical separation between the plan generation engine and the encoding of the planning domain. The planning process itself involves the computationally expensive task of reasoning about actions (NP-complete in general [2]) that forces "efficient" planning systems to make heuristic choices amongst alternatives during planning. Here we are interested in *goal choice*, the problem of choosing which outstanding goal to achieve during goal directed planning. This is an important problem in general, since plans that achieve goals separately can interact in harmful ways when merged to form a plan to achieve goals conjunctively. Consequently, much planning research has focussed on the issues of dealing with goal interactions in ways that avoid so-called goal clobbering [18, 19, 4].

Increasingly within machine learning research there has been an exploration of techniques to acquire control rules for improving, automatically, the efficiency of a general planner when applied to a particular domain. Techniques are chiefly aimed at creating domain-dependent heuristics for guiding planning choices, and can be grouped into one of two categories: those that learn from the analysis of planning traces after a planning session and those that perform a static or off-line analysis of the planner's domain theory.

Both approaches have advantages and limitations. Training-based techniques (e.g. [17]) can benefit from a training set's bias if the set is characteristic of the kind of problems to be solved in performance mode, but it may not be straightforward to acquire or construct such a set, and the order of training problems may be critical. Work on the PRODIGY/EBL system [15] has highlighted yet further problems: training examples can encourage the production of overly specific control rules and in certain circumstances acquired control knowledge can degrade performance (this is the well known problem of *utility* [14]).

Research into static analysis of planning domains to increase on-line planning efficiency includes the goal ordering technique of Cheng and Irani [10], acquisition of abstraction hierarchies in ALPINE [11] and control rule learning in STATIC [1]. As pointed out by Etzioni in [1], following [8], static analysis can be usefully viewed as a kind of partial evaluation: a general planning engine takes both a planning domain and a particular problem as input. If the planning domain is input in advance, the planner can be "partially evaluated" to produce a more efficient planner for application to input problems.

In this paper we present PRECEDE, a novel method of acquiring heuristic goal orders that can be used to determine goal choice during goal directed planning. Calculation of the orders is based on an analysis of domain operators that outputs necessary interactions between pairs of predicates in a planning domain. It is a static form of learning which is completely independent of on-line planning resources since all analysis is performed a priori. When used to select goals during plan generation these orders linearise parts of the planning process, as the problem of choice between competing goals in a goal-directed search is greatly reduced. Since the method is problem-independent all learning is *off-line* and it therefore incurs negligible overhead during planning, unlike many learning techniques which encounter the utility problem.

The paper is organised as follows: we start by introducing notation that will be used in the definition of PRECEDE, and by familiarising the reader with a planning domain that will be used as an example. This is followed in section 3 by a discussion of *inconsistency* which will be used in the definition of PRECEDE goal orders. Our emphasis in this paper is to give a precise specification of the PRECEDE goal orders and this, along with a worked example, is the subject of section 4. The method is fully implemented and has been empirically tested, and in section 5 we briefly discuss some of the results from several test configurations. Finally in section 6 we compare PRECEDE with related static learning methods.

2 Notation and Example World

We assume a model of planning along the lines of that adopted by Lifschitz [12]. In particular, a planner is input with a state-based domain theory containing operators, initial state, goals and state axioms. Furthermore, the planner's reasoning is based on the *default persistence* and *closed world* assumptions.

We will let OS denote the set of parameterised operators for such a domain. Assume each operator A in OS has a conjunction of predicates for pre-conditions, denoted $A.pre$, and sets of predicates representing positive and negative effects, that are written $A.add$ and $A.del$ respectively. We define what it means for an operator $A \in OS$ to be a possible *achiever* for some predicate x as follows:

$$achiever(A, x, u) \Leftrightarrow \exists\, y \in A.add : mgu(x, y, u)$$

where $mgu(x, y, u)$ means that predicates x and y unify under the most general unifier u. A possible *clobberer* A for predicate x is similarly defined:

$$clobberer(A, x, u) \Leftrightarrow \exists\, y \in A.del : mgu(x, y, u)$$

As an illustration we will use examples taken from a "warehouse" domain that consists of a warehouse with a driver, truck, crane and crates that can be moved to different locations in the warehouse. This domain was originally used to test training based learning approaches in planning, and is fully specified in [16]. In this case the domain is structured using several sorts, including spaces, crates and shelves which strongly type the domain predicates. The domain is depicted in figure 1 below.

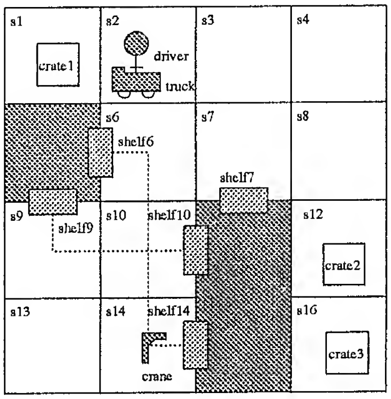


Fig. 1. The warehouse planning domain: *WW*

3 Formulation of PRECEDE

Assume that a conjunction of predicates i can match with a planning state K, termed $match(i, K)$, if and only if there is some legal binding s of constants to variables in i which results in every ground predicate of i_s evaluating to true under the interpretation given by K. For example, every predicate in u_s, where

$$u = at(crate1, S_i) \wedge at(crate2, S_j) \wedge \neg(S_i = S_j)$$

and

$$s = \{s1/S_i, s12/S_j\}$$

evaluates to true under the interpretation given by the state of the example planning domain in figure 1.

PRECEDE requires a set of negative state invariants, called IN, as part of the input domain, specifying what *cannot* be true within a valid state. These invariants can be either user supplied (an assumption we make here) or they can (at least in part) be automatically generated from operators in the domain [16]. In either mode they have the added advantage of focusing on the validity of the domain model, and possibly highlighting inconsistencies which were not intended by the domain modeller.

If any member i of set IN matches with a state K, then K is said to be invalid. More formally,

$$\exists\, i \in IN : match(i, K) \Rightarrow invalid(K)$$

As an example, the set IN for the domain in figure 1 contains 30 negative invariants, which include the following two:

$$in\_truck \wedge driver\_on\_floor(S)$$
$$at(C, S_i) \wedge at(C, S_j) \wedge \neg(S_i = S_j)$$

The first states that the *driver* cannot be simultaneously in the *truck* and on the floor. The second captures the constraint that any *crate* (denoted by variable C) cannot be simultaneously at two different spaces (where S_i, S_j are variables of sort *space*).

The formulation of PRECEDE requires us to reason with state expressions generalised to include variables ranging over sorts, such as that found in preconditions of operator schema. Such a conjunction of predicates u can also be invalid with respect to a domain definition, if every state that u can match with is invalid as defined above. We call such an expression *inconsistent*:

$$(\forall\, K : match(u, K) \Rightarrow invalid(K)) \Rightarrow inconsistent(u)$$

In other words, if an expression u matches with the same or a subset of the states as a member of IN, then u is inconsistent.

4 PRECEDE: Goal Order Definition

Now we will define an ordering on predicates in the domain definition, to be used to constrain planning search, using the following basic definition. Assume x and y are predicates within some conjunction of predicates u (where u is typically a pre-condition expression for some operator). Then we say that x must be achieved before y if for any A which is a possible achiever of x, A cannot also achieve y, and either its the case that A clobbers y or y forms an inconsistent logical expression when conjoined to A's pre-conditions. Formally, this is written:

$$
\begin{aligned}
&\mathit{before}(x, y) \Leftrightarrow \\
&\quad \forall A, t : \mathit{achiever}(A, x, t) \Rightarrow \\
&\qquad \neg (\exists\, r : \mathit{achiever}(A_t, y_t, r)) \wedge \\
&\qquad \exists\, s : \mathit{clobberer}(A_t, y_t, s) \vee \mathit{inconsistent}(A.\mathit{pre}_t \wedge y_t)
\end{aligned}
$$

If a pair of predicates (x, y) satisfy this relation then x must be achieved before y in any plan that makes both predicates true in the same valid planning state. This means that it is necessary to tackle the *second* predicate before the *first* during goal-directed planning since the "direction" proceeds backwards from the goal to the initial state.

As an example, we will show that for $\mathit{truck\_at}(S_i)$ and $\mathit{driver\_on\_floor}(S_j)$ (where S_i and S_j are variables of sort *space*) the following condition is true.

$$\mathit{before}(\mathit{truck\_at}(S_i), \mathit{driver\_on\_floor}(S_j))$$

There is only one operator that is an achiever for $\mathit{truck\_at}(S_i)$ and that is $\mathit{drive}(S, S_i)$ with

$$\mathit{drive}(X, Y).\mathit{add} = \{\mathit{truck\_at}(Y), \mathit{clear}(X)\}$$

and mgu $t = \{S_i/Y\}$. Since the only other goal that this operator achieves cannot unify with $\mathit{driver\_on\_floor}(S_j)$ it can be seen that no achiever of $\mathit{truck\_at}(S_i)$ also achieves $\mathit{driver\_on\_floor}(S_j)$.

Now we can show that the pre-conditions of the achiever of $\mathit{truck\_at}(S_i)$ are *inconsistent* with the set containing $\mathit{driver\_on\_floor}(S_j)$. For the only possible achiever of $\mathit{truck\_at}(S_i)$, we have

$$\mathit{drive}(X, Y).\mathit{pre} = \mathit{in\_truck} \wedge \mathit{truck\_at}(X) \wedge \mathit{clear}(Y)$$

Thus

$$A.\mathit{pre}_t \wedge y_t = \mathit{in\_truck} \wedge \mathit{truck\_at}(X) \wedge \mathit{clear}(S_i) \wedge \mathit{driver\_on\_floor}(S_j)$$

which is inconsistent, as for any state K that this can match with, its also true that $\mathit{match}(i, K)$, where

$$i = \mathit{in\_truck} \wedge \mathit{driver\_on\_floor}(S)$$

is the first member of IN for the warehouse domain that was discussed earlier. Thus $\mathit{before}(\mathit{truck\_at}(S_i), \mathit{driver\_on\_floor}(S_j))$ is true.

The *before* relation could be used to build up a global set of goal orders for different planning domains in a number of ways. In fact PRECEDE analyses all and only those pairs of predicates that appear together in some operator pre-conditions, and obtains a set of predicate pairs that obey the *before* relation using a procedure which implements the definition above. This relation is then augmented by carefully generating its transitive closure, based on the assumption that:

$$before(x, z1) \wedge before(z2, y) \wedge mgu(z1, z2, s) \Rightarrow before(x_s, y_s)$$

for some domain predicates x, y, $z1$ and $z2$. An alternative approach is test *all* pairs of uninstantiated predicates in a domain with the *before* relation. The main problem here lies in the consideration of possible co-designation relations between variables of the same sort occurring in the predicates. As observed in a series of experiments with the STATIC system [13] comparing uninstantiated pairs of predicates means that all the possible co-designation relations between the different predicates must be considered. So for example, if the predicates are $at(C_i, S_i)$ and $truck\_loaded(C_j, S_j)$ (where C_i and C_j are variables of sort *crate* and S_i and S_j are of sort *space*) then they must be considered under each of the following co-designation constraints:

$$C_i = C_j \wedge S_i = S_j$$
$$C_i \neq C_j \wedge S_i = S_j$$
$$C_i = C_j \wedge S_i \neq S_j$$
$$C_i \neq C_j \wedge S_i \neq S_j$$

In fact, for each pair of predicates the number of possible co-designation relationships is 2^n (where n is the number of "slots" of the same sort), which is unacceptable in the event of large n.

PRECEDEs method means that only co-designation relationships that might be encountered when applying operators during plan generation need to be considered. As a consequence, considerably less computation is required to generate this goal order and all the goal orders that are identified will be "useful" since they will arise whenever backchaining occurs from the operator they appear in.

As an example, consider again $at(C_i, S_i)$ and $truck\_loaded(C_j, S_j)$. These appear in the operator $drive\_load(C, S_x, S_y)$ in the form $truck\_loaded(C, S_x)$ and $at(C, S_x)$. This is equivalent to the single co-designation relationship

$$C_i = C_j \wedge S_i = S_j$$

reflecting the fact that to drive a *truck* loaded with a *crate*, the *crate* must be at the same location as the *truck*. There is an obvious saving here since the three other co-designation relations need not be considered.

5 Discussion of Empirical Evaluation

PRECEDE has been used in conjunction with a goal directed planner called MEA, within FM, a planning and learning system that allows users to "mix and match" different plan generation engines, learning components and modelled planning domains [16]. The implementation of PRECEDE is input a planning domain (operators, state axioms and a set of negative state invariants) and outputs an ordering relation on predicates (called a "PRECEDE goal order") as discussed in the section above. The impact of PRECEDE goal orders on planner performance has been tested on a number of planning domains, on a number of differently controlled problem samples and in comparison with other learned control rules. Below we show its performance on two augmented, traditional STRIPS-worlds [6]: the first with a single robot and the second with two robots working together. The third graph shows its performance on the warehouse world. The implementation itself takes approximately 10 cpu seconds to acquire the PRECEDE goal order for the robot worlds, and 100 cpu seconds to generate the goal order for the warehouse world (figures are from a SUN Sparc running compiled Prolog). We feel that this is acceptable since the processing need only be performed once for a given planning domain.

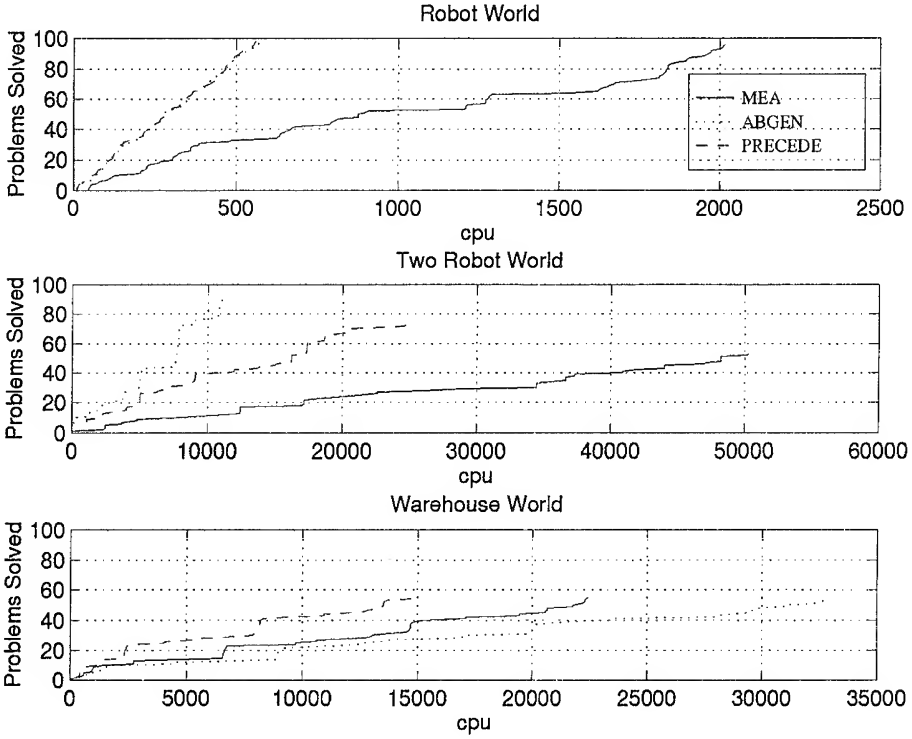


Fig. 2. Empirical Results

The results of some of these tests are summarised in figure 2. The accumulated cpu time for solving randomly generated planning problems by MEA is shown as the continuous line in the graph. MEA is a relatively efficient planner in that it makes certain linearity assumptions during plan generation, and it has a set of base heuristics to help search. Most importantly, it inputs operators in a *structured* form so that preconditions are already marked as *environmental*, *filters* and *achievable*, and effects are structured into those that are primary and those that are side-effects.

For the results shown by the dashed line, MEA was adapted so that it uses PRECEDE goal orders as preference heuristics, returning to its weak heuristics in the absence of orders. Finally, the dotted line represents the accumulated cpu on a version of MEA adapted so that it uses goal orders as preference heuristics generated by a routine called ABGEN. This orders predicates in the same form as PRECEDE, but the algorithm is based on Knoblock's ALPINE algorithm [11] (although his orders were originally used to generate an abstraction hierarchy rather than as straightforward ordering heuristics during planning). The difference in the static analysis is that ABGEN builds up orders based on *possible* interactions, whereas PRECEDE is based on necessary interactions.

For the robot planning domain we can see that PRECEDE and ABGEN both result in an overall speed up of a factor of about 4 when compared with the basic planner MEA, and, in fact, both techniques return similar predicate orderings. For the domain containing two robots both PRECEDE and ABGEN show a marked improvement over MEA. ABGEN is superior to PRECEDE because it is able to generate a more useful goal ordering in this loosely coupled domain. This occurs because the introduction of an additional robot makes goal achievement more flexible, resulting in fewer necessary orders. On the other hand, the warehouse world is an example of a domain where operators are tightly coupled, and whereas PRECEDE finds a useful ordering which improves the performance of MEA, ABGEN performs less well. This is due to an overwhelming number of possible interactions which, when merged together, introduce cycles which cause a collapse in the ordering structure (this can be viewed as an instance of the "collapsing problem" referred to by [7]). These tests suggest that PRECEDE will result in an improvement on a wide range of application domains, and will be particularly suitable to strongly coupled domains. Also, the results are especially promising as MEA is already a relatively efficient planner.

6 Related Work

PRECEDE falls into the category of machine learning techniques that create domain dependent heuristics for planning through a process of static analysis of a planners domain theory. In this section we compare it with some of the other techniques in this class.

ALPINE [11] is a method for automatically generating abstraction hierarchies for planning. Unlike PRECEDE, the hierarchies are problem specific (they are generated at run time for input planning goals) and they are based on possible interactions between goals whereas PRECEDE's goal orders are based on necessary interactions. Nevertheless there is a correspondence between PRECEDE goal orders and ALPINE abstraction hierarchies that guarantees that whenever $before(x, y)$ is true, for any two predicates x and y, then y cannot be at a higher level in an ALPINE abstraction hierarchy than x (interested readers are referred to the proof in [16]).

Cheng and Irani [9, 10] describe a technique for augmenting conjunctive planning goals and computing a partial order over the goal which, they argue, eliminates the need to backtrack over protected goals during plan generation. The orders are generated by back propagating the input goals through the operators to identify *consistent* sets of predicates that must necessarily hold to apply the operators. This is in contrast to PRECEDE which reasons about inconsistent sets of predicates during its analysis. Another difference between the two methods is that the goal ordering technique of Cheng and Irani is problem specific since goal orders are generated at run time for an input conjunctive planning goal whereas PRECEDE is problem independent.

An early example of static analysis can be found in Dawson and Siklóssy's REFLECT system [3]. It compiled a set of "incompatible assertions", pairs of predicates that cannot be simultaneously true in a valid state of a modelled planning domain, which are similar to the negative state invariants IN, that feature in PRECEDE. The difference between the two methods lies in the way that this information is used. Whereas REFLECT uses incompatible assertions during plan generation to select partial plans with consistent outstanding preconditions, PRECEDE uses IN to identify goal interactions and form goal orders independently of on-line planning.

STATIC [5] is also a technique that acquires control rules for planning, through static analysis of a model of a planning domain. It identifies necessary interactions between goals and uses these as the basis for control rule acquisition. The analysis performed by STATIC and PRECEDE is markedly different, however. STATIC constructs a series of graphs (one for each predicate in a planning domain) that represent the relationship between goals and operators in a given domain model. These graphs are then traversed to identify interactions between goals. In contrast, PRECEDE identifies interactions based on the *before* condition, which in turn utilises negative state invariants.

7 Conclusions

In this paper we have built up a declarative specification of PRECEDE, a novel off-line compilation method that generates a goal order which is based on necessary interactions between predicates. It uses information about inconsistent states of a domain, in the form of a set of negative state invariants which can be either supplied by the domain modeller or automatically generated from the set of domain operators OS. PRECEDE has been tested against and in combination with other learning methods using several different controlled experiments and over several planning domains. Here we described some of the results of those experiments, and gave a brief comparison of PRECEDE with related techniques.

In future work we plan to integrate PRECEDE with other static learning methods, such as macro operator generation [16].

References

1. O. Ezioni . Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62, 1993.
2. T. Bylander. Complexity Results for Planning. In *Twelfth International Joint Conference on Artificial Intelligence*, 1991.
3. C. Dawson and L. Siklóssy. The Role of Preprocessing in Problem Solving Systems. In *Fifth International Joint Conference on Artificial Intelligence*, 1977.
4. M. Drummond and K. Currie. Goal Ordering in Partially Ordered Plans. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.
5. O. Etzioni. Why PRODIGY/EBL Works. In *Eighth National Conference on Artificial Intelligence*, 1990.
6. R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2, 1971.
7. E. Fink and Q. Yang. Automatically Abstracting the Effects of Operators. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, 1992.
8. F. Van Harmelen and A. Bundy. Explanation-Based Generalisation = Partial Evaluation. *Artificial Intelligence*, 36:401 – 412, 1988.
9. K. B. Irani and J. Cheng. Subgoal Ordering and Goal Augmentation for Heuristic Problem Solving. In *Tenth International Joint Conference on Artificial Intelligence*, 1987.
10. K. B. Irani and J. Cheng. Ordering Problem Subgoals. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.
11. C. Knoblock. *Automatically Generating Abstractions for Problem Solving.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.
12. V. Lifschitz. On the Semantics of STRIPS. In *Proc. 1986 workshop: Reasoning about actions and plans*, 1986.
13. M. A. Perez and O. Ezioni . DYNAMIC: A new role for training problems in EBL. Technical Report CMU-CS-92-124, School of Computer Science, Carnegie Mellon University, 1992.
14. S. Minton. Quantitative Results concerning the utility of Explanation-Based Learning. In *Seventh National Conference on Artificial Intelligence*, pages 564 – 569, 1988.
15. S. Minton, J. G. Carbonell, C. A. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gill. Explanation-Based Learning: A Problem Solving Perspective. *Artificial Intelligence*, 40:63 – 118, 1989.
16. J. M. Porteous. *Compilation-Based Performance Improvement for Generative Planners.* PhD thesis, Department of Computer Science, The City University, 1993.
17. T. L. McCluskey. Explanation-based and Similarity-Based Learning. In *Proceedings of the 4th European Working Session on Learning.* Pitman, 1989.
18. A. Tate. Project Planning Using a Hierarchical Non-linear Planner. Technical Report Department of Artificial Intelligence memo 75, University of Edinburgh, 1977.
19. D. Wilkins. *Practical Planning: Extending the Classical A.I. Paradigm.* Addison-Wesley, 1988.

Learning Problem-Oriented Decision Structures from Decision Rules: The AQDT-2 System

Ryszard S. Michalski and Ibrahim F. Imam

Center for Artificial Intelligence
George Mason University
Fairfax, VA. 22030
{michalski, iimam}@aic.gmu.edu

ABSTRACT

A decision structure is an acyclic graph that specifies an order of tests to be applied to an object (or a situation) to arrive at a decision about that object. and serves as a simple and powerful tool for organizing a decision process. This paper proposes a methodology for learning decision structures that are oriented toward specific decision making situations. The methodology consists of two phases: 1—determining and storing declarative rules describing the decision process, 2—deriving on-line a decision structure from the rules. The first step is performed by an expert or by an AQ-based inductive learning program that learns decision rules from examples of decisions (AQ15 or AQ17). The second step transforms the decision rules to a decision structure that is most suitable for the given decision making situation. The system, AQDT-2, implementing the second step, has been applied to a problem in construction engineering. In the experiments, AQDT-2 outperformed all other programs applied to the same problem in terms of the accuracy and the simplicity of the generated decision structures.

Key words: machine learning, inductive learning, decision structures, decision rules, attribute selection.

1 Introduction

The main step in the development of an advisory system for decision making is the creation of a knowledge structure that characterizes the decision making process. A simple and effective tool for describing decision processes is a *decision structure*, which is an acyclic graph that specifies an order of tests to be applied to an object (or a situation) to arrive at a decision about that object. The nodes of the structure are assigned individual tests (which may correspond to a single attribute, a function of attributes, or a relation), the branches are assigned possible test outcomes (or ranges of outcomes), and the leaves are assigned one specific decision or a set of candidate decisions (with corresponding probabilities), or an *undetermined* decision. A decision structure reduces to a familiar decision tree, when each node is assigned a single attribute and has at most one parent, the branches from each node are assigned single values of that attribute, and leaves are assigned single, definite decisions. Thus, the problem of generating a decision structure is a generalization of the problem of generating a decision tree.

Decision trees are typically generated from a set of examples of decisions. The essential characteristic of any such method is the *attribute selection criterion* used for choosing attributes to be assigned to the nodes of the decision tree being built. Such criteria include the entropy reduction [13, 14], the gini index of diversity [5], and others [6, 7, 12].

A decision tree/decision structure can be an effective tool for describing a decision process, as long as all the required tests can be measured, and the decision making situations it was designed for remain constant. Problems arise when these assumptions do not hold. For example, in some situations measuring certain attributes may be difficult or costly. In such situations it is desirable to reformulate the decision structure so that the “inexpensive” attributes are evaluated first (are assigned to the nodes close to the root), and the "expensive" attributes are evaluated only if necessary (are assigned to the nodes far away from the root). If an attribute cannot be measured at all, it is useful to either modify the structure so that it does not contain that attribute, or—when this is impossible—to indicate alternative candidate decisions and their probabilities. A restructuring is also desirable, if there is a significant change in the frequency of occurrence of different decisions.

A restructuring of a decision structure (or a tree) in order to suit new requirements is usually quite difficult. This is because a decision structure is a procedural knowledge representation, which imposes an evaluation order on the tests. In contrast, no evaluation order is imposed by a declarative representation, such as a set of decision rules. Tests (conditions) of rules can be evaluated in any order. Thus, for a given set of rules, one can usually build a large number of logically equivalent decision structures (trees), which differ in the test ordering. Due to the lack of “order constraints,” a declarative representation (rules) is much easier to modify to adapt to different situations than a procedural one (a decision structure or a tree). On the other hand, to apply decision rules to make a decision, one needs to decide in which order tests are evaluated, and thus, needs a decision structure.

An attractive solution of these opposite requirements is to acquire and store knowledge in a declarative form, and transform it to a decision structure when it is needed for decision making. This method allows one to create a decision structure that is most appropriate in a given decision making situation. Because the number of decision rules per decision class is usually small (much smaller than the number of training examples per class), generating a decision structure from decision rules can potentially be performed much faster than by generation from training examples (methods generating decision trees from examples, are described in 5, 13, 14, 15). Thus, this process could be done “on line,” without any delay noticeable to the user. Such “virtual” decision structures are easy to tailor to any given decision making situation.

This approach allows one to generate a decision structure that avoids or delays evaluating an attribute that is difficult to measure. Initial research on this approach, and the first system implementation, AQDT-1, are described in [8]. This paper presents a new version of the system, called AQDT-2. The new system generates a goal-oriented decision structure from decision rules learned by either rule learning system AQ15 [11] or system AQ17, which has extensive constructive induction capabilities [2]. AQDT-2 has several new features, including: 1) a method for utilizing new attributes, not present in the original data, derived by constructive induction, 2) a method for controlling the degree of generalization needed during the development of the decision structure, 3) two new attribute selection criteria, 4) a new method for combining different attribute selection criteria, 5) the ability to generate “unknown” nodes in situations, when there is insufficient information for generating a complete decisiòn structure, 6) the ability to learn decision structures from "discriminant" decision rules, as well as "characteristic" rules, and 7) the ability to provide the most likely decision from a set of candidate decisions when the decision process is prematurely stopped due to the inability to measure an attribute associated with some node. New features of AQDT-2 are demonstrated in an experiment on determining a decision structure for choosing wind bracings for tall buildings [1]. The results briefly illustrate how the system tailors decision structures to different decision making situations.

2 The AQDT-2 Method

This section describes the AQDT-2 method for building a decision structure from decision rules. The method for building a single-parent decision structure is similar to that used in standard methods of building a decision tree from examples. The major difference is that it assigns tests (attributes) to the nodes using criteria based on the *properties of the decision rules*, rather than *statistics characterizing the coverage* of training examples. Other differences are that the branches may be assigned an internal disjunction of values (not only a single value, as in a typical decision tree), and leaves may be assigned a set of alternative decisions with probabilities. Also, the tests can be attributes or names standing for logical or mathematical expressions that involve several attributes or variables. In the following, we use the terms "test" and "attribute" interchangeably (to distinguish between an attribute and a name standing for an expression, the latter is called a *constructed* attribute).

At each step, the method chooses the test from an available set of tests that has the highest *utility* (see below) for the given set of decision rules. This test is assigned to the node. The branches stemming from this node are assigned test values, or disjoint groups of values (in the form of logical disjunction, if such occur in the rules; subsumed groups of values are removed). Each branch is associated with a reduced set of rules, determined by removing conditions in which the selected attribute assumes value(s) assigned to this branch. If all rules in the reduced ruleset indicate the same decision class, a leaf node is created and assigned this decision class. The process continues until all nodes are leaf nodes. If it is not possible to reduce further the ruleset (because some attribute is declared as unavailable (infinite cost), then the leaf is assigned a set of candidate decisions with associated probabilities (see Sec. 4.2).

The test (attribute) utility is a combination of one or more of the following elementary criteria: 1) *disjointness*, which captures the effectiveness of the test in discriminating among decision rules for different decision classes, 2) *importance,* which determines the importance of a test in the rules, 3) *value distribution*, which characterizes the distribution of the test importance over its of values, and 4) *dominance*, which measures the test presence in the rules. These criteria are defined below.

Cost : The cost of a test expresses the effort or cost needed to measure or apply the test.

Disjointness. The disjointness of a test is defined as the sum of the *class disjointness*—the disjointness of the test for each decision class. Suppose decision classes are C_1, C_2,..., C_m, and decision rulesets for these classes have been determined. Given test A, let V_1, V_2,....,V_m, denote sets of the values (outcomes) of A that are present in rulesets for classes C_1, C_2,..., C_m, respectively. If a ruleset for some class, say, C_t contains a rule that does not involve test A, than V_t is the set of all possible values of A (the domain of A).

Definition 1. The *degree of class disjointness*, D(A, C_i) *of test A for the ruleset of class C_i*, is the sum of the degrees of disjointness, D(A, C_i, C_j), between the ruleset for C_i and rulesets for C_j, j=1, 2,...m, $j \neq i$. The degree of disjointness between the ruleset for C_i and the ruleset for C_j is defined by:

$$D(A, Ci, Cj) = \begin{cases} 0, & \text{if } Vi \subseteq Vj \\ 1, & \text{if } Vi \supset Vj \\ 2, & \text{if } Vi \cap Vj \neq \phi \text{ or } Vi \text{ or } Vj \\ 3, & \text{if } Vi \cap Vj = \phi \end{cases} \quad (1)$$

where ϕ denotes the empty set.

Definition 2. The *disjointness* of the test A for evaluating a given set of decision rules is the sum of the degrees of class disjointness of each decision class:

$$\text{Disjointness}(A) = \sum_{i=1}^{m} D(A, C_i) \qquad \text{where} \qquad D(A, C_i) = \sum_{i=1,\, i\neq j}^{m} D(A, C_i, C_j) \quad (2)$$

The disjointness of a test ranges from 0, when the test values in rulesets of different classes are all the same, to 3*m*(m-1), when every ruleset of a given class contains a different set of the test values.

Importance. The second elementary criterion, the importance of a test, is based on the *importance score* (IS), introduced in [9]. In the obtained rules, each test is assigned a "score" that represents the total number of training examples, which are covered by the rules involving this test. Decision rules learned by an AQ learning program are accompanied with information on their strength. Rule strength is characterized by *t-weight* and *u-weight.* The *t*-weight (*total-weight*) of a rule for some class is the number of examples of that class covered by the rule. The u-weight (*unique-weight*) of a rule for some class is the number of examples of that class covered only by this rule. The importance score of a test is the aggregation of the total-weights of all rules that contain that test in their condition part. Given a set of decision rules for m decision classes $C_1,..,C_m$, and involving n tests $A_1,..,A_n$, and the number of rules associated with class C_i is denoted by "r_i". The importance score is defined as follow.

Definition 3. The *importance score*, $IS(A_j)$, of the test A_j is determined by:

$$IS(Aj) = \sum_{i=1}^{m} IS(Aj, Ci) \qquad \text{where} \qquad IS(Aj, Ci.) = \sum_{k=1}^{r_i} Rik(Aj) \quad (3)$$

and R_{ik}, the weight of a test A_j in the rule R_k of class C_i is given by:

$$R_{ik}(A_j) = \begin{cases} t - \text{weight} & \text{if } A_j \text{ belongs to rule } R_{ik} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where i=1,..,n; ik=1,..,ri; j=1,..,m.

Value distribution. The third elementary criterion, value distribution, concerns the number of legal values of tests. Given two tests with equal importance scores, this criterion prefers the test with the smaller number of legal values. Experiments have shown that this criterion is especially useful when using discriminant decision rules.

Definition 4. A *value distribution*, $VD(A_j)$ of a test A_j is defined by:

$$VD(A_j) = IS(A_j) / v_j \quad (5)$$

where "v" is the number of legal values of Aj.

Dominance. The fourth elementary criterion, dominance, prefers tests that appear in large number of rules, as this indicates their high relevance for discriminating among ruleset of given decision classes. Since some conditions in the rules have values linked by internal disjunction, counting such rules directly would not reflect properly their relevance. Therefore, for computing the dominance, the rules are counted as if they were converted to rules that do not have internal disjunction. Such a conversion is done by "multiplying out" the condition parts of the rules containing internal disjunction. For example, the condition part [x3=1 v 3]&[x4=1] is "multiplied out" to two rules with condition parts [x3=1]&[x4=1] and [x3=3]&[x4=1].

The above criteria are combined into one general test measure using the "lexicographic evaluation functional with tolerances" (LEF) [10]. LEF is a list of some or all of the above elementary criteria, each associated with a "tolerance threshold" in percentage. The criteria are applied to tests in the order defined by LEF. A test passes to the next criterion only if it scores on the previous criterion within the range defined by the tolerance (from the top value). The default LEF is:

<Cost $\tau 1$; Disjointness, $\tau 2$; Importance, $\tau 3$;Value distr., $\tau 4$; Dominance, $\tau 5$> (6)

where $\tau 1$, $\tau 2$, $\tau 3$ and $\tau 4$ are tolerance thresholds (in percentages); their default values are 0. The default value of the cost of each test is 1.

3 An Illustration of the Method

This section illustrates the method by applying it to a problem of learning a decision structure for determining the structural quality of a tall building design. The quality of the design is partitioned into four classes: high (c1), medium (c2), low (c3), and infeasible (c4). Each example is characterized by seven attributes: number of stories (x1), bay length (x2), wind intensity (x3), number of joints (x4), number of bays (x5), number of vertical trusses (x6), and number of horizontal trusses (x7). The data consisted of 335 examples, of which 220 (66%) were randomly selected to serve as training examples, and 115 (34%) were used for testing the obtained decision structure.

In the first phase, training examples were used to determine a set of decision rules. This was done by the program AQ15c (Michalski, et al. 1986; a new version written in "C" was used). Figure 1 shows decision rules obtained by AQ15c.

These rules were then used by AQDT-2 to determine a decision structure. Table 1 presents values of the four elementary criteria for each attribute occurring in the rules, for the step of determining the root of the decision structure. For each class, the row marked "Values" lists values occurring in the ruleset for this class. For evaluating the disjointness of an attribute, say A, each rule in the ruleset that does not contain attribute A is assumed to contain an additional condition [A= a v b ...], where a, b, ... are all legal values of A.

Assuming the default LEF, attribute x6 was chosen for the root (since its disjointness is the single highest and all other attributes are beyond the tolerance threshold, no other attributes are considered). Branches stemming from the root are marked by values x6 (in general, it could be groups of values), according to the way they occur in the decision rules (groups subsumed by other groups are removed [8]. The branches are assigned subsets of the rules containing these values. The process repeats for a branch until all rules assigned to each branch are of the same class. That class is then assigned to the leaf.

Figure 2 presents a decision structure determined by AQDT-2 from decision rules in Figure 1 (using the default LEF). The structure was evaluated on the testing examples. The prediction accuracy was 88.7% (102 testing examples were classified correctly and 13 miss-classified). For comparison, the program C4.5 for learning decision trees from examples was also applied to this same problem [15]. The experiment was done with C4.5 using the default window setting (maximum of 20% the number of examples and twice the square root the number of examples), and set the number of trials set to one.

Decision class C1

1	[x1=1][x6=1][x2=1,2][x3=1,2][x4=1,3][x5=1,2][x7=1..3]	(t :18, u :18)
2	[x1=3][x2=1][x3=1][x5=1][x6=1][x4=1,3][x7=1,3,4]	(t :3, u : 3)
3	[x1=5][x2=2][x3=2][x5=2][x4=3][x6=1][x7=2,3]	(t :2, u : 2)
4	[x1=1][x6=1][x2=2][x3=1,2][x4=3][x5=1,2][x7=4]	(t :2, u : 2)
5	[x1=3][x2=1][x4=1][x6=1][x7=1][x3=2][x5=1,2]	(t :2, u : 2)
6	[x1=1][x3=1][x6=1][x2=2][x4=1,3][x7=1,3][x5=3]	(t :2, u : 2)
7	[x1=2][x5=2][x2=1][x6=1][x3=1,2][x4=3][x7=4]	(t :2, u : 2)

Decision class C2

1	[x1=2..4][x2=1,2][x3=1,2][x4=3][x5=2,3][x6=1][x7=2,3]	(t :28, u :19)
2	[x1=2..4][x2=2][x3=1,2][x4=3][x5=1,2][x6=1][x7=3,4]	(t :17, u : 6)
3	[x1=2,4][x2=1,2][x3=1,2][x4=3][x5=1][x6=1][x7=3,4]	(t :10, u : 4)
4	[x1=1,3,5][x2=1,2][x3=1,2][x4=3][x5=3][x6=1][x7=2,4]	(t :10, u : 2)
5	[x1=3,5][x2=1,2][x3=1,2][x4=3][x5=2,3][x6=1][x7=1,4]	(t: 9, u : 4)
6	[x1=2][x2=1,2][x3=1,2][x5=1,2,3][x4=1][x6=1][x7=1]	(t: 7, u : 6)
7	[x1=3,4][x2=2][x3=2][x4=1,3][x5=1,3][x6=1][x7=1,2]	(t: 6, u : 4)
8	[x1=3,5][x2=2][x3=1][x7=1][x4=1,2][x5=1,2,3][x6=1,3]	(t: 5, u : 5)
9	[x1=1][x2=1][x6=1][x3=1,2][x4=3][x5=1,2][x7=4]	(t: 4, u : 4)
10	[x1=1][x5=1][x2=2][x4=2][x6=2][x3=1,2][x7=1..3]	(t: 4, u : 4)
11	[x1=1,2][x2=1][x6=1][x3=1,2][x4=1,3][x5=3][x7=1,4]	(t: 4, u : 2)

Decision class C3

1	[x1=2..5][x2=1,2][x3=1,2][x7=1..4][x4=1,2][x5=1,3][x6=2,4]	(t :41, u :32)
2	[x1=1..4][x2=1,2][x3=1,2][x4=2][x5=2][x6=2,3][x7=2..4]	(t :27, u :20)
3	[x1=1,3][x2=1][x3=1,2][x7=1..4][x4=2][x5=1,2][x6=2,3]	(t :19, u : 6)
4	[x1=1,2,4][x2=1,2][x3=1,2][x4=2][x5=2,3][x6=3,4][x7=1]	(t :13, u : 8)
5	[x1=5][x2=2][x4=2][x5=2][x3=1,2][x6=3][x7=2..4]	(t: 5, u : 5)

Decision class C4

1	[x1=5][x2=2][x3=2][x4=1,3][x5=1][x6=1][x7=1..4]	(t: 4, u : 4)
2	[x1=5][x2=2][x3=1][x5=1][x6=1][x4=3][x7=3]	(t: 1, u : 1)

Figure 1: Decision rules determined by AQ15c from the wind bracing data.

Class		Attributes						
		x1	x2	x3	x4	x5	x6	x7
C1	Values	1,2,3,5	1,2	1,2	1,3	1,2,3	1	1..4
	Class disjointness	1	1	0	2	1	3	0
C2	Values	1..5	1,2	1,2	1,2,3	1,2,3	1,2,3	1..4
	Class disjointness	2	1	0	3	1	4	0
C3	Values	1..5	1,2	1,2	1,2	1,2,3	2,3,4	1..4
	Class disjointness	2	1	0	4	1	8	0
C4	Values	5	1	1,2	1,3	1	1	1..4
	Class disjointness	0	0	0	2	0	3	0
Attribute Disjointness		5	3	0	11	3	18	3
Attribute Importance		245	82	25	245	233	245	181
Attribute value distribution		49	41	13	82	78	61	45
Attribute Dominance		45	34	42	33	40	30	54

Table 1: Values of selection criteria for each attribute for the wind bracing problem.

The decision tree learned by C4.5 had a lower predictive accuracy, namely 84.3% (97 testing examples were classified correctly and 18 were miss-classified). The C4.5 tree was also considerably more complex (it had 17 nodes and 43 leaves in contrast to 5 nodes and 9 leaves in the decision structure).

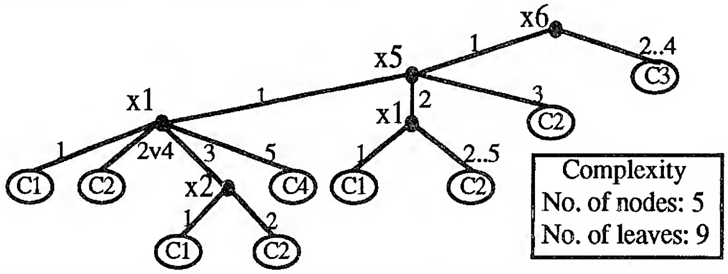


Figure 2: A decision structure determined by AQDT-2 for the wind bracing problem.

4 Tailoring Decision Structure to the Decision Making Situation

4.1 Handling the Attribute Cost

As described in Sec. 2, the LEF criterion enables the system to take into consideration the cost of measuring tests (attributes) in developing a decision structure. In the default setting of LEF, the test's cost is the first criterion, and its tolerance is 0. This means that only the least expensive attributes pass to the next step of evaluation involving other elementary criteria. If an attribute has high cost, or is impossible to measure (has an infinite cost), the LEF chooses another, "cheaper" attribute, if possible. Figure 3 shows a decision structure obtained from the rules in Figure 1 under the condition that x1 cannot be measured. Leaves marked ? represent situations in which a definite decision cannot be made without knowing x1. This "incomplete" decision tree was tested on 115 testing examples from which value of x1 was removed. The decision structure classified 71 examples correctly, 14 incorrectly, and 30 were assigned the "?" (indefinite) decision. Next subsection describes how the "?" leafs can be replaced by sets of candidate decisions with their corresponding probability distribution.

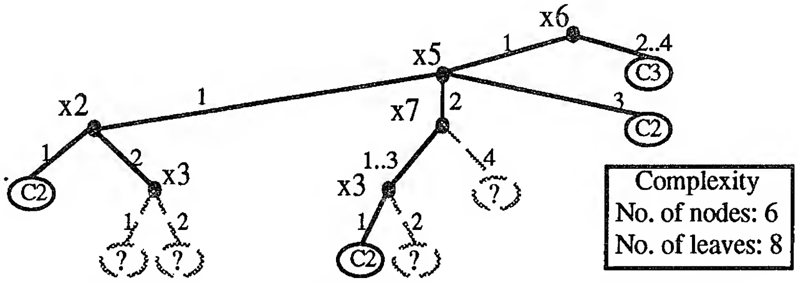


Figure 3: A decision structure that does not contain attribute x1.

4.2 Assigning Decision Under Insufficient Information

In decision making situations in which some attribute or attributes cannot be measured, the system may not be able to assign a definite decision for some cases. In the previous subsection, such cases were identified by leaves denoted by "?". If no more information can be obtained, but a decision has to be made, it is useful to know the probability distribution for different candidate decisions. The most probable decision is then chosen. The probability distribution can be estimated from the class frequency at the given node. Let us consider a node in the structure connected to the root by the sequence b1, b2,.., bk of attribute values. Let P(Ci I b1, b2, bk) denote the conditional probability of class Ci, i=1,2,...,m, at that node, given that an example to be classified has attribute values assigned to branches b1, b2, ..bk in the decision structure. Using Bayesian formula, we have:

$$P(Ci \mid b1, \ldots, bk) = P(Ci) * P(b1, \ldots, bk \mid Ci) / P(b1, \ldots, bk) \qquad (7)$$

where P(Ci) the apriori probability of class Ci ; P(b1, ... , bk I Ci) is the probability that the example has attribute values b1,..., bk, given that it represents decision class Ci. To approximate these probabilities, let us suppose that w_i is the number of training examples of Class Ci that passed the tests leading to this node), and tw_i — the total number of training examples of Class Ci. Let us use these numbers to estimate the probabilities in (7). Assuming that the apriori class probabilities correspond to the frequency of training examples from different classes, we have:

$$P(Ci) = tw_i \,/ \sum_{j=1}^{m} tw_j \qquad (8)$$

$$P(b1, \ldots, bk \mid Ci) = w_i \,/\, tw_i \qquad (9)$$

$$P(b1, \ldots, bk) = \sum_{j=1}^{m} w_j \,/ \sum_{j=1}^{m} tw_j \qquad (10)$$

By substituting (8), 9 and 10 in 7, we obtain:

$$P(Ci \mid b1, \ldots, bk) = w_i \,/ \sum_{j=1}^{m} w_j \qquad (11)$$

Figure 4 presents a decision structure from Figure 3 in which ? leaves were assigned candidate decisions with decision class probability estimates. Let us consider node x2. The example frequencies were: w1=31, tw1=45; w2=11, tw2=139; w3=0, tw3=169, and w4=5, tw4=5. Using equation (11), the probability estimates for classes C1, C2, C3 and C4 under node x2 can be approximated as P(C1)= . 66, p(C2) = .23, P(C3)=0 and P(C4) =.11.

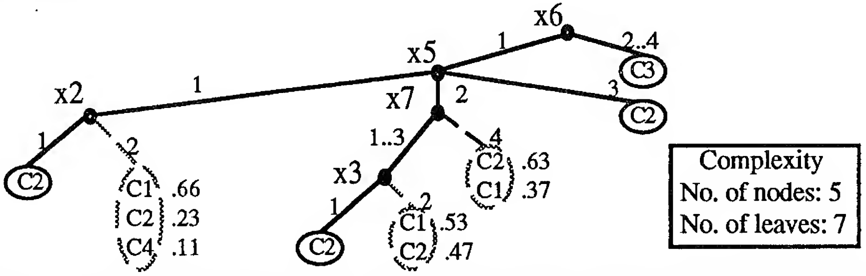


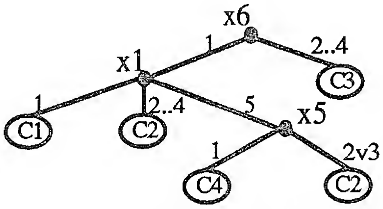
Figure 4: A decision structure without X1 with candidate decisions assigned to ? leafs

A related method for handling the problem of unavailability of an attribute is described by Quinlan [15]. His method, however, assigns probabilities to the given decision tree, without first trying to restructure it appropriately to a given decision making situation (in this case, to avoid measuring x1). AQTD-2 assigns probabilities only after it first created a decision structure that suits the given decision making situation.

4.3 Coping with noise in training data

The proposed methodology can be easily extended to handle the problem of learning from noisy training data. This is done based on ideas of rule truncation described in [2, 7]. When noise is expected in the training data, the decision rules with t-weight below

a certain threshold (reflecting the expected noise level) are removed. The rule truncation method seems to result in more accurate decision structures than decision tree pruning method because truncation decisions are solely based on the importance of the given rule or condition for the decision making, regardless of their evaluation order (unlike the decision tree pruning, which can only prunes attributes within a subtree, and thus cannot freely chose attributes to prune).
Figure 6 shows the decision structure resulting from decision rules in Figure 1, which were truncated under the assumption of the 10% noise level (this means that the rules whose combined t-weight represented 10% or less coverage of the training examples in a given class were removed). The predictive accuracy of this decision structure on the testing data was 88% (in contrast to 89% for the decision structure in Figure 2).



Complexity
No. of nodes: 3
No. of leaves: 5

Figure 6: A decision structure determined from rules in Figure 2 under the assumption of 10% classification error in the training data.

5. Conclusion

The paper introduced the concept of a decision structure, and described a methodology for efficiently determining a single-parent structure from a set of decision rules. A decision structure is an acyclic graph that defines a conditional order of tests for arriving at a classification decision for a given object or situation. Having higher expressive power than the familiar decision tree, a decision structure is able to represent some decision processes in a much simpler way than a decision tree.

The proposed methodology advocates to store the decision knowledge in the declarative form of decision rules, which are determined by induction from examples or by an expert. A decision structure is generated "on line," when is needed, and in the form most suitable for the given decision making situation (i.e., a class of cases of interest). A criticism may be leveled against this methodology that in order to determine a decision structure from examples, it requires to go through two levels of processing, while there exist methods that produce decision trees efficiently and directly from examples. Putting aside the issue that decision structures are more general than decision trees, it is argued that this methodology has many advantages that fully justify it. Main advantages include: 1) decision structures produced by the methods in the experiments conducted had higher predictive accuracy and were simpler (sometimes very significantly) than decision trees produced from the same data 2) decision structures produced from rules can be easily tailored to the given decision making situation, e.g., can avoid measuring expensive attributes, or have them in the lowest parts of the structure, 3) by storing decision knowledge in a declarative form of modular decision rules, the methodology makes it easy to modify decision knowledge to account for new facts or changing conditions, 4) the process of deriving a decision structure from a set of rules is very fast and efficient, because number of rules per class is usually much smaller than the number of examples per class, and 5) the presented method produces decision structures, whose nodes can be original attributes, or constructed attributes that extend the original knowledge representation (this is due to the application of constructive induction programs AQ17-DCI and AQ17-DCI). The price for these advantages is that the system has to generate decision rules first, and from them can create decision structures. In the method proposed, this first phase is done by an AQ algorithm-based rule learning method. While past implementations of AQ-based methods were computationally complex, the most recent implementation is very fast [16]; thus the generating decision rules phase can be done quite efficiently.

The current method has a number of limitations, and several issues need to be investigated further. First of all, there is need for more testing of the method. Although the experiments conducted so far have produced more accurate and simpler decision structures than decision trees obtained in a standard way for the same input data, more experiments are needed to arrive at conclusive results. A mathematical analysis of the method has not been performed and is highly desirable. The current method generates only single-parent decision structures (every node has only one parent, like in a decision tree). Extending the method to generate full-fledged decision structures (in which a node can have several parents) will make it more powerful. It will enable the method to represent much more simply decision processes that are difficult to represent by a decision tree (e.g., a symmetric logical function). The decision structures produced by the method are usually more general than the decision rules from which they were created (they may assign decisions to cases that rules could not classify). Further research is needed to determine the relationship between the certainty of decision rules and the certainty of decision structures derived from them. The AQ-based program allows to generate both characteristic and discriminant decision rules (Michalski, 1983). There is need to investigate the advantages and disadvantages of generating decision structures from different types of rules.

ACKNOWLEDGMENTS

The authors thank Eric Bloedorn for his insightful comments. This research was conducted in the Center for Artificial Intelligence at George Mason University. The Center's research is supported in part by the Office of Naval Research under grant No. N00014-91-J-1351, in part by the National Science Foundation under grant No. IRI-9020266, in part by the Defense Advanced Research Projects Agency under the grant No. N00014-91-J-1854, administered by the Office of Naval Research, and the grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research.

REFERENCES

1. **Arciszewski, T, Bloedorn, E., Michalski, R., Mustafa, M., and Wnek, J.,** "Constructive Induction in Structural Design", Report of Machine Learning and Inference Laboratory, MLI-92-7, Center for AI, George Mason University., 1992.

2. **Bergadano, F., Matwin, S., Michalski R. S. and Zhang, J.,** "Learning Two-tiered Descriptions of Flexible Concepts: The POSEIDON System," *Machine Learning*, Vol. 8, No. 1, pp. 5-43, January 1992.

3. **Bloedorn, E., Wnek, J., Michalski, R.S., and Kaufman, K.,** "AQ17: A Multistrategy Learning System: The Method and User's Guide", Report of Machine Learning and Inference Laboratory, MLI-93-12, Center for AI, George Mason University. 1993.

4. **Bohanec, M. and Bratko, I.**, "Trading Accuracy for Simplicity in Decision Trees", Machine Learning Journal, Vol. 15, No. 3, Kluwer Academic Publishers, 1994.

5. **Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J.**, "Classification and Regression Structures", Belmont, California: Wadsworth Int. Group, 1984.

6. **Cestnik, B. & Bratko, I.**, "On Estimating Probabilities in Structure Pruning", *Proceeding of EWSL 91*, (pp. 138-150) Porto, Portugal, March 6-8, 1991.

7. **Cestnik, B. & Karalic, A.**, "The Estimation of Probabilities in Attribute Selection Measures for Decision Structure Induction" in *Proceeding of the European Summer School on Machine Learning*, July 22-31, Priory Corsendonk, Belgium, 1991.

8. **Imam, I.F. and Michalski, R.S.**, "Learning Decision Structures from Decision Rules: A method and initial results from a comparative study", in *Journal of Intelligent Information Systems JIIS,* Vol. 2, No. 3, pp. 279-304, Kerschberg, L., Ras, Z., & Zemankova, M. (Eds.), Kluwer Academic Pub., MA, 1993.

9. **Imam, I.F., Michalski, R.S. and Kerschberg, L.**, "Discovering Attribute Dependence in Databases by Integrating Symbolic Learning and Statistical Analysis Techniques", *Proceeding of the First International Workshop on Knowledge Discovery in Database*, Washington, D.C., July, 11-12, 1993.

10. **Michalski, R.S.**, "AQVAL/1-Computer Implementation of a Variable-Valued Logic System VL1 and Examples of its Application to Pattern Recognition", *Proceeding of the First International Joint Conference on Pattern Recognition*, (pp. 3-17), Washington, DC, October 30- November 1, 1973.

11. **Michalski, R.S., Mozetic, I., Hong, J. & Lavrac, N.**, "The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains", *Proceedings of AAAI-86*, (pp. 1041-1045), Philadelphia, PA, 1986.

12. **Mingers, J.**, "An Empirical Comparison of selection Measures for Decision-Structure Induction", *Machine Learning*, Vol. 3, No. 3, (pp. 319-342), Kluwer Academic Publishers, 1989a.

13. **Quinlan, J.R.**, "Discovering Rules By Induction from Large Collections of Examples", in D. Michie (Edr), *Expert Systems in the Microelectronic Age*, Edinburgh University Press, 1979.

14. **Quinlan, J.R.**, "Learning efficient classification procedures and their application to chess end games" in R.S. Michalski, J.G. Carbonell and T.M. Mitchell, (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Los Altos: Morgan Kaufmann, 1983.

15. **Quinlan, J. R.** "Probabilistic decision structures," in Y. Kodratoff and R.S. Michalski (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. III*, San Mateo, CA, Morgan Kaufmann Publishers, (pp. 63-111), June, 1990.

16. **Wnek, J.**, "A Fast Re-Implementation of the AQ-based Inductive Learning Program for Large Datasets: AQ15c," *Reports of Machine Learning and Inference Laboratory*, MLI 94-3, Center for Artificial Intelligence, George Mason University, 1994 (to appear).

Towards Full Automation of the Discovery of Heuristics in a Nuclear Engineering Project: Integration With a Neural Information Language

Ephraim Nissan and Hava Siegelmann,[1] Alex Galperin and Shuky Kimhi[2]

[1] Department of Mathematics and Computer Science,
Bar-Ilan University, Ramat-Gan, Israel
[2] Department of Nuclear Engineering,
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract. FUELCON is an expert system in nuclear engineering. The practitioner of nuclear fuel management uses it to generate configurations of reactor core refueling. The domain expert, on the other hand, employs the system to discover new heuristics for the previous task, based on performance during previous iterations in the same session. Expert use involves a manual phase of revising the ruleset. We expose the building blocks of the design of a new version, that incorporates a neural component to carry out the revision. It uses the system's previous performance it for adaptation and learning better rules. for adaptation and learning better rules. The neural component is based on a particular language and schema for symbolic to recurrent-analog conversion, called NEL, and on reinforcement learning for the adaptation.

1 Nuclear Plants

Nuclear plants include one or more units, i.e. *reactors.* Each reactor has a *core,* where nuclear fuel generates energy through a process of *fission,* i.e. of splitting of the "heavy" atoms of uranium. Fuel units, called *fuel-assemblies,*, are packages, 350 cm long, of 200 to 250 cilindrical rods —made of uranium dioxide— whose diameter is 1 cm. The section of an assembly is square, as assemblies have to be inserted each into a square case in a planar grid, which is the surface or a section of a reactor core. By this grid, the *core geometry* is schematized. It is symmetric, so reasoning for the application described in this paper —*fuel allocation*— is done on just a one-eighth slice of the core schema. The positions in the grid are typified by both **(a)** the geometry, and **(b)** other factors. E.g., **a**$'$: Is the position on a symmetry diagonal? **a**$''$: How many positions separate the position considered from the center of the core? **b**$'$: Is a *control rod* (i.e. a device related to pressurization) located upon the given position?

Among the fuel rods, liquid coolant, pumped from below the assembly, circulates. It usually is pressurized water. Pressure is needed to prevent the water from boiling. There are steam generators on the borders of the core: they act as heat exchangers. Just part of the thermal energy thus produced can be exploited as electric power: e.g. about 3000 thermal megawatts would yield about 1000 electric megawatts. Such is the power supply generated by a typical *pressurized-water reactor (PWR).* A single fuel-rod generates about 80 kilowatts of thermal energy. One fuel-assembly generates about 20 megawatts.

There are two areas in *fuel management,* within nuclear engineering: **(i)** the management of fuel as kept in store, and **(ii)** the allocation of fuel to be burned in the reactor core during a certain temporal interval between two *refueling points.* Area **(ii)** is the task of our *FUELCON* expert system.

2 Nuclear Fuel Reload Design

Most kinds of nuclear plants practice *annual reload:* the reactor is shut down for a few weeks, and, during this *downtime period,* the reactor is inspected, and a specialized engineer, called the *in-core fuel manager,* has to design a reallocation of fuel units into the core of the reactor. The new allocation is intended for effective power generation during the next *operation cycle.* Once the engineer has provided the design, it is validated: standard simulations are mandated by regulations, for *safety* purposes; moreover, quality is checked in terms of *effectiveness.* Upon approval, the design is implemented: actual refueling takes place. There exist different kinds of *fuel.* One criterion of classification is how many times the fuel unit has already been used. (This is reflected in a *cumulated burnup* degree.) Indeed, once a reactor is shut down, not all of the fuel from the activity period just concluded is dumped. Just such fuel that has been used during three cycles is disposed of, and replaced with *fresh fuel.* Instead, fuel that was used once *(once-burned),* or twice *(twice-burned),* is put back into the *fuel inventory,* that also includes some fresh fuel. Fuel is costly. The assemblies in the inventory at the plant are the pool from which to pick one fuel-unit per position in the grid. This is one constraint, concerning resources, and is of *economic* nature. The geometry of the core is a *structural* constraint; each reactor has its own individuality. There are *safety* constraints: e.g. fresh fuel must not be placed in the central region of the grid, lest overheating would make cooling ineffective. (Less dramatically than a disaster, local overheating in some region of the core could damage some fuel-unit, making it unfit for reuse.)

The length of the annual operation cycles at plants is variable; so are requirements of power supply. Longer active periods are desirable, economically. And, shortening the current downtime period —by making reload design quicker— can save millions of dollars.

3 Reload-Design Automation, and FUELCON

There exist tools for assisting the practitioner in designing fuel reload [4]. There are tools that just help with either computations or graphics, to perform single steps in the design procedure, whereas the overall procedure is manual. Tools from another class embody some optimization method from operations research, as customized for the task. Algorithms adopted there, do not transparently reflect the intuitive rules of thumb of the human expert. *Observability* in the space of solutions is very reduced, so such tools just seek one solution that is admissible, and, if possible, locally optimal. Generally, one solution for the *in-core fuel management problem* is one full configuration of core-position/fuel-assembly allocation, for the entire core. It will be referred to as *configuration* for short.

There have been one or two attempts to apply AI, but these did not depart from the conception of manual design as seeking just one solution [5]. Typically, a given solution, known —e.g. from some published case-study— to have suited a different situation, is adapted to the new situation. This is done by means of a series of *shuffles,* i.e. by switching the positions of two (or more) fuel-assemblies. Conceptually, such a procedure is depth-first search, with shuffles constituting backtracking. Rules check whether the candidate shuffle is admissible. Thereafter, simulation has to check whether the new solution is admissible for the problem at hand, i.e. for the given reactor after a given operation cycle.

Our own expert system, *FUELCON,* is a major contribution to the application domain, where it got a good reception. [2] is an AI-oriented description. Ambitions, with FUELCON, are of global optimization. FUELCON generates not one, but hundreds of alternative configurations per session. A heuristic ruleset finds a good position for a given fuel-assembly, and this is done recursively, until the entire core is filled with all of the available fuel units. Basically, search is breadth-first. There is no backtracking. The output of the generation module is fed to *NOXER,* a simulator. The numeric results of NOXER are visualized. Each configuration is displayed as a dot in the cartesian plane of two parameters: x is *boron concentration at EOC* (i.e. at the *end of the cycle* concluded by the current downtime period), and has to be maximized. As to y, it is *power peaking:* $y < \bar{y}$ defines the *admissibility window,* whose "southwestern" corner is at the origin of the coordinates. Instead, configurations above $\bar{y}$ are unsafe. As FUELCON produced hundreds of alternative solutions, a "cloud" of solutions is displayed in the plane. Sometimes, none is safe. Or, then, some are, but they are not satisfactory, because $x < \bar{x}$. The interesting region is $y < \bar{y} \wedge x > \bar{x}$. Whereas the practitioner may be satisfied with being able to pick one configuration that happens to be in a good region, the domain expert, when using FUELCON, may wish to move the "cloud" of solutions into a "southeast" direction, in order to get several configurations that are both safe and very efficient. S/he may wish to test new heuristics, or to concentrate search (by focusing: *zooming)* on a cluster of interesting configurations, previously unknown to researchers in the domain. To do so, s/he manually revises the ruleset. This is the step we automate by means of a neural component that we designed, based on the *NEL* language and symbolic-to-neural conversion schema, as defined by Siegelmann [6].

Figure 1 depicts, on the left, a fuel assembly and a symmetric slice of the core. The figure shows the flow of information through the architecture of our system. Two sample displays of NOXER are given.

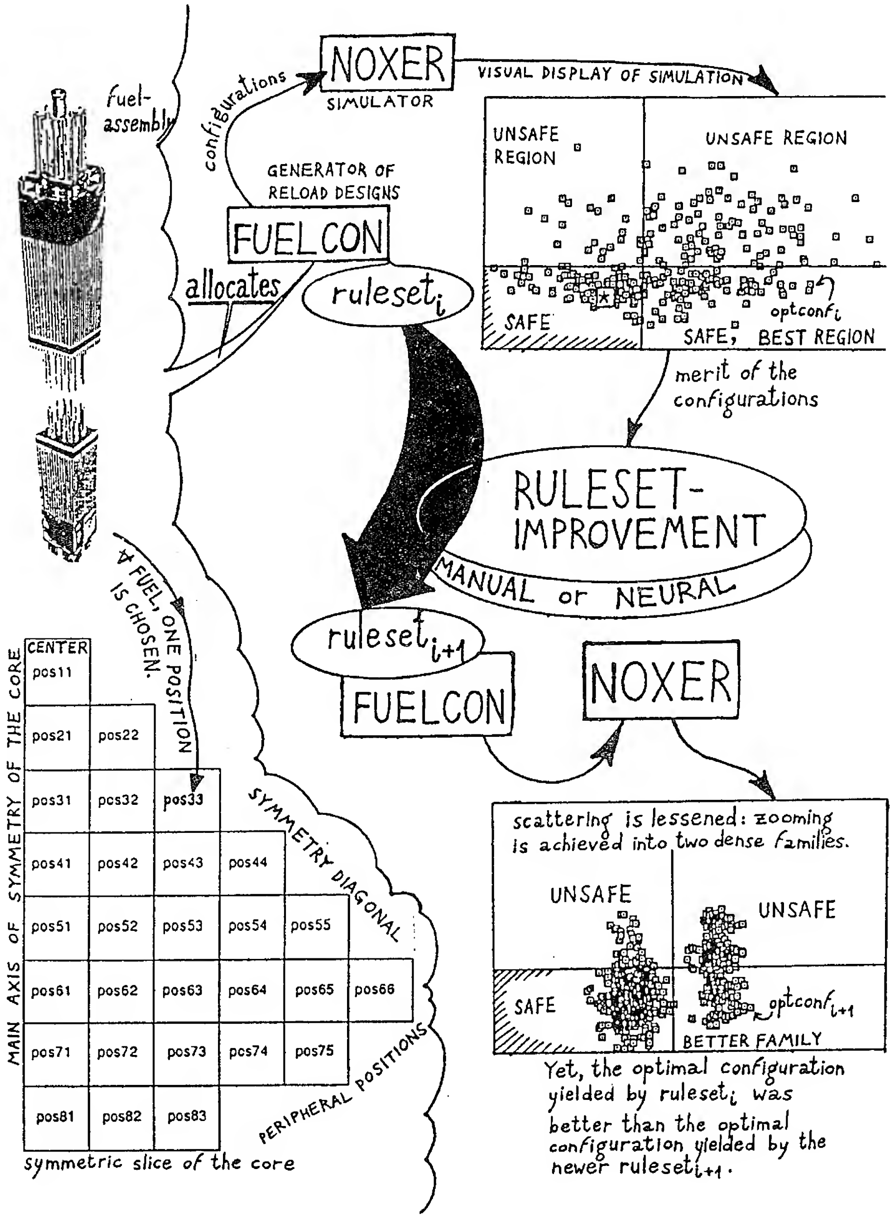


Fig. 1. A depiction of what the expert system does.

4 An Example of Ruleset, and Manual Revision

FUELCON generates a tree, whose leaves are alternative full configurations. Each level in the tree corresponds to a given fuel-assembly, picked from the inventory, and that generally can be placed in alternative positions in the core. The choice of the position is guided by a ruleset. Part of the rules are *elimination rules:* such *Don't* rules are never revised, if dictated by safety concerns. Or, then, they may reflect the expert's wish to circumscribe search to an interesting direction. The rest of the rules are a positive prescription; they are *preference rules,* subject to revision, and typifying the regions in the search-space searched during a given session. The *database* of FUELCON includes **(i)** a sub-database of fuel types and units, **(ii)** a component that describes the geometry and the features of the reactor core, and **(iii)** the collection of (partial or final) configurations being gradually constructed (internal nodes in the tree, or leaves). Let us exemplify a configuration-generating ruleset. We omit the causes or goals that suggested the rules: these are not explicit, in FUELCON. In one real-case study, the following initial ruleset was used:

1. *Don't load any fresh fuel-assembly in any such position that its distance from the center of the core is shorter that the distance of POS44 therefrom.* (Distance is integer).
2. *Don't load a fresh assembly in such a position that is adjacent to another position where there is another assembly of the same kind, except when one of those two positions is in a corner position, that is, except when one of those two positions is adjacent along two of its sides to the reflector* (i.e. the surrounding cooling water).
3. *Don't place any twice-burned assembly in any of the positions of the eighth row and of positions POS74, POS75, and POS66.*
4. *Don't load a twice-burned assembly in such a position that is adjacent to another position where there is another twice-burned assembly, if the positions considered are comprised in rows 5 to 8 in the core.*
5. *Don't load any such twice-burned assembly that has a very high value of cumulated burnup (over 20500 MWd/t), adjacently to a position containing a twice-burned assembly.*
6. *Don't load any twice-burned assembly in any position belonging to any of rows 2, 3, or 4, if more than one position adjacent thereto does contain a twice-burned assembly.*
7. *If it is a twice-burned assembly that is currently being considered, then choose for it (from amongst those positions that were not forbidden by Rules 1 to 6) that position whose distance from the center of the core is minimal.*
8. *If it is a once-burned assembly that is currently being considered, then choose for it (from amongst those positions that were not forbidden by Rules 1 to 6) that position whose distance from the center of the core is minimal.*

Rules 1 to 6 are elimination rules. Rules 7 and 8 are not mandatory from the physical viewpoint, but are preference rules intended for pruning, and are enacted last. Once results were obtained and simulated, the domain experts spotted which positions in the core configurations caused the maximal power density, which led to modifications in the ruleset, as follows. Rule 2, about adjacent fresh fuel-assemblies, was modified to allow adjacency even when one of the positions is adjacent to the reflector on just one side. One more rule was formulated that is concerned with once-burned assemblies. It is an elimination rule: **Rule 9:** *Don't place any once-burned assembly in any position adjacently to a fresh assembly, if the position considered for loading the once-burned assembly belongs to any of rows 2, 3 or 4 in the core.* Indeed, the analysis of the configurations generated by the ruleset as originally formulated, indicated that the situation excluded by the new Rule 9 caused an increase of local power density, in the region involved, to a range of values between 1.4 and 1.5. As the threshold assumed is 1.4, it was suitable to have a specific rule preventing this kind of situations.

Now, the generator was run again; simulation by NOXER led to further manual revision, of both the ruleset, and the given loading sequence of assemblies. (It is given, just like the rules.) We claim that the heuristics of revision can be automated. The rest of the paper exposes the basics in terms of which we designed the neural component to carry this out. A language and a general procedure are outlined. A long forthcoming paper, [3], exposes and exemplifies the customization for the given ruleset.

5 Rules as in Neural Network Format, and NEL in general

In neural computation, *reinforcement learning* is typified by this schema: input is fed to the network, it computes some output and a "score" is returned from the environment, implying how well it computed. The network adapts in an effort to receive a better score [1]. The coding schema exposed hereby is the platform for encoding the rules of FUELCON as a neural network, for automated improvement. (It does not provide the reverse, net-to-rules conversion.) The design envisages having the rules automatically refined to achieve a better score. Elimination rules, intended to ensure safety, must not be changed. Preference rules may be reconsidered.

Let us see how to translate the rules —both mandatory and optional— and the related part of the engine into a network. A network operates generally as follows. There are N neurons. At each tick of the clock, all neurons are updated with new values. Thus, a network step consists of a parallel execution of N assignments. When simulating the program on a network, some of the neurons represent variables, and others represent the flow control and other parts of the program. Actually, about half of the neurons in the network are related to the program itself and the other half constitutes the program counters. We first translate the rules into a recurrent net with a particular piece-wise activation function, $\sigma(x) := 0$ if $x < 0$, $\sigma(x) := x$ if $0 \le x \le 1$, and $\sigma(x) := 1$ if $x > 1$. When the translation is done, the activation function at each neuron is converted into sigmoidal form: $\sigma(x) = 1/(1+e^{-x})$. This yields a network amenable to known results on reinforcement learning.

We first consider two examples of compilation:

Example 1. Let M and N assume values in $[0,1]$ and B be a Boolean expression. The conditional statement **If** (B) **then** $x = M$ **else** $x = N$ is simulated by $x_1(t) = \sigma(N-B)$, by $x_2(t) = \sigma(M+B-1)$, and by $x_3(t+1) = \sigma(x_1(t)+x_2(t))$, where x_3 provides the value of the statement. □

Example 2. Each assignment statement s_k has associated substatement counters $c_k^{(1)}$ and $c_k^{(2)}$, and in case it appears in a parallel block, it also has an associated flag counter c_k^{par} . A substitution $x = y$ (where the values of x and y are in the range $[0..1]$) is thus executed as follows:

$$x_1^+ = \sigma(x - c_k^{(1)}) \qquad x_2+ \ = \sigma(y + c_k^{(1)} - 1)$$

$$x^+ = \sigma(x_1 + x_2) \qquad c_k^{(2)+} = \sigma(c_k^{(1)}) \ .$$

If s_k appears in a sequential block, $c_{k+1}^{(1)+} = \sigma(c_k^{(2)})$ updates the next statement counter. If s_k appears inside a parallel block with other p statements, its flag neuron and the next statement counter are updated, by, respectively, $c_k^{\text{par}\,+} = \sigma(c_k^2 + c_k^{\text{par}} - c_m^{(1)})$ and $c_m^{(1)+} = \sigma(c_k^{\text{par}} + \sum_{j=1}^{p} c_j^{\text{par}} - p)$. □

The second example has demonstrated that the process of translating rules and programs into nets is not trivial one. In general, tasks may be composed of a large number of interrelated subtasks. The entire task may thus be highly complex, and designing an appropriate network becomes infeasible. A high-level language is called for. We use *NEL* (for **NE***ural* **L***anguage)* [6]. NEL combines features of both Pascal and Lisp and is rich enough to express any computer algorithm or rule-based system. NEL programs (or rulesets) are compiled to an equivalent network that compute just like them and even require the same amount of time. In this sense, NEL is not only a programming language but also a translation schema. Note that the network itself should consist of analog neurons only and does not allow for any threshold neuron, so that the resulting network complies with continous adaptation and learning algorithms. In this paper, we provide such an interface. NEL is a procedural, modular, parallel, very expressive language. There is a wide range of possible data types, operators, and statements. We are going to consider the implementation of a sample of these, from the compiler design.

NEL programs have to be compiled into neural networks. A network operates generally in the following manner: There are N neurons. At each tick of the clock, all neurons are updated with new values. Thus, a network step consists of a parallel execution of N assignments.

When simulating the program on a network, some of the neurons represent variables, and others represent the flow control and other parts of the program.

Actually, about half of the neurons in the network are related to the program itself and the other half constitutes the program counters. More specifically, each statement is associated with a special neuron, called the "statement counter" neuron. These neurons take Boolean (i.e., binary) values only. When a statement counter neuron is True, the statement is executed. Note that several statement counters may assume the value True simultaneously. The next example demonstrates the compilation of various data types:

Example 3. Let c be a counter and x_1, x_2 be real variables. (The operator Dec(c) decrements the counter, and Dec(0) = 0.) Consider the sequence: $x_1 = x_2 = 1$; **read**(c) ; **Repeat Dec**(c) ; $x_1 = \frac{1}{2}x_1$; $x_2 = \frac{1}{7}x_2$ **Until Iszero**(c) .

A counter variable may assume any natural value or 0. In the network compilation, to each variable c of the counter type, there is associated an activation variable z whose value is $z = 1 - 2^{-c}$. The counter operations are implemented as follows. Dec*(c)* is implemented as $z^+ = \sigma(2z - 1)$. Inc*(c)* is implemented as $z^+ = \sigma(\frac{1}{2}z + \frac{1}{2})$. Iszero*(c)* is implemented as $z^+ = \sigma(1 - 2z)$. As to real variables, they are implemented as real values in the range $[0, 1]$.

We want a network to simulate the above program, assuming initial values $x'_1 = x'_2 = 1$, and $c' = 1 - 2^{-c}$. Here is the net. **Computing** $\mathbf{x_1}$: ${x'_1}^+ = \sigma(\frac{1}{2}x'_1)$. **Computing** $\mathbf{x_2}$: ${x'_2}^+ = \sigma(\frac{1}{7}x'_2)$. **Counter:** $c'^+ = \sigma(2c' - 1)$, $s^+ = \sigma(3 - 4c' - s')$, and $s'^+ = \sigma(3 - 4c')$. As to the **Output,** it is: (x'_1, x'_2, s) . Here s takes the role of a binary validation neuron which is set to 1 once, implying that the output information is ready. □

A more detailed description follows of how to compile NEL into a network. First, let us consider **net representation of data types and expressions.** Each variable of any data type, except for records and arrays, is represented via one neuron in the network. We provide as example the implementation of a few data types. Boolean values are represented via the numbers $\{0, 1\}$. The **logical operations,** with the respective net-emulations, are as follows. **Not***(x)* is implemented as $\sigma(1 - x)$; **Or**(x_1, x_2) as $\sigma(x_1 + x_2)$; and **And**(x_1, x_2) as $\sigma(x_1 + x_2 - 1)$. **Relational operations** are defined in a straightforward manner: $x > y$ is $\sigma(x - y)$, $x \geq y$ is $\sigma(x - y + 1)$, and $x \neq y$ is $\sigma(\sigma(x - y) + \sigma(y - x))$. As to **scalars,** there may be up to Fix different elements in each scalar type. Assume a scalar type with n elements $\{0, 1, \ldots (n - 1)\}$. The ith element is represented as scalar $(i, n) \hookrightarrow (2i + 1)/2n$. **Order operations** are implemented as follows: **Pred**(x) as $\sigma(x - \frac{1}{n})$, **Succ**(x) as $\sigma(x + \frac{1}{n})$, and **Ord**(x) as $\sigma(xn - \frac{1}{2})$. Using this representation, it follows from the fact that n is always bounded by the constant Fix, that the relational operators of scalar type always involve numbers separated by a distance of at least $\frac{1}{2\text{Fix}}$.

As to **counters,** we have: counter(n) $\hookrightarrow .\overbrace{11\ldots1}^{n}$. I.e. a counter whose value is n is represented as $(1 - 2^{-n})$. Let us see **operations on counters. Inc**(x) is implemented as $\sigma(\frac{1}{2}(x + 1))$; **Dec**(x) as $\sigma(2x - 1)$; and **Iszero**(x) as $\sigma(1 - 2x)$.

A **list** of T is encoded via a number between 0 and 1. Assume T is general scalar type with cardinality n. The list $l = a_1 a_2 \cdots a_k$ is represented by $l = a_1 a_2 \cdots a_k \hookrightarrow \sum_{i=1}^{k}((2a_i + 1)/(2n)^i)$. A special case for Boolean $n = 2$ is $\sum_{i=1}^{k}((2a_i + 1)/4^i)$. We can write the general case also as $\sum_{i=1}^{k}((\text{ scalar } (a_i, n))/(2n)^{i-1})$. This is a number in the range $[0, 1)$. If $\mathbf{Ord}(\mathbf{Car}(l)) = i$ $(i = 0, \ldots, (n-1))$, then the value of the encoding ranges in $[p, q]$, where $p = (2i+1)/2n$, and $q = (2i+2)/2n$. The second value in the list further restricts the available range. If $\mathbf{Ord}(\mathbf{Car}(l)) = i$ and $\mathbf{Ord}(\mathbf{Cadr}(l)) = j$, then the value of the encoding ranges in $[p + r, p + s]$, where $r = (2j+1)/4n^2$ and $s = (2j+2)/4n^2$. In summary, not every value in $[0, 1]$ appears. The set of possible values is not continuous and has "holes". Such a set of values "with holes" is a Cantor set. Its self-similar structure means that bit (base $2n$) shifts preserve the "holes." The advantage of this approach is that there is never the need to distinguish among two very close numbers in order to read the **ccar** of the list. Now, if a list l is represented via the number x, then: $\mathbf{Car}(l)$ is implemented as $\sigma(\frac{1}{2n} + \frac{1}{n}(\sigma(2nx - 2) + \sigma(2nx - 4) + \cdots + \sigma(2nx - (2n - 2))))$; $\mathbf{Cdr}(l)$ as $\sigma((2nx - 1 - 2(\sigma(2nx - 2) + \sigma(2nx - 4) + \cdots + \sigma(2nx - (2n - 2)))))$; $\mathbf{Cons}(e, l)$ as $\sigma(\frac{x}{2n} + e)$; and $\mathbf{Isnull}(x)$ as $\sigma(1 - 2nx)$. **Stacks** are handled like lists. **Sets, records,** and **arrays** are amenable to such representations.

Let us consider **statements.** We skip, here, assignment, goto jumps, input/output, subroutine calls (of kinds subprogram and function). A **parallel block** is translated into parallel execution of the individual statements. In the following NEL program, we associate statement counters with the associated statements, writing them to the left of the statements. Consider the block: c_k : **Parbegin** $(c_{1,k}^{1}, \ldots, c_{1,k}^{l_{1,k}}, c_{1,k}^{\text{par}})$: statement$_1$; ... ; $(c_{n,k}^{1}, \ldots, c_{n,k}^{l_{n,k}}, c_{n,k}^{\text{par}})$: statement$_n$; **Parend** followed by another block labeled c_{k+1} , etc. When c_k is set to 1, all substatement counters are set simultaneously to 1. The parallel block may end only when statement$_1$ to statement$_n$ finish their executions. Generally, the associated statement counters are updated as follows: **Parallel** $\forall i$: $c_{i,k}^{1+} = \sigma(c_k)$; $c_{i,k}^{j+} = \sigma(c_{i,k}^{j-1})$ where $j = 2 \ldots l_{i,k}$; $c_{i,k}^{\text{par}+} = \sigma(c_{i,k}^{l_{i,k}} + c_{i,k}^{\text{par}} - c_{k+1})$; $c_{k+1}^{+} = \sigma(\sum_{j=1}^{n} c_{i,k}^{\text{par}} - (n-1))$.

As to **serial blocks,** they are implemented via parallel blocks: **Begin** statement$_1$; ... ; statement$_n$ **End** where we have to use the statement counters $c_1^1 \ldots c_1^{l_1}, c_2^1 \ldots c_2^{l_2}, \ldots c_n^1 \ldots c_n^{l_n}$.

We assume a control line that has the value 1 once. The operations are then: $c = \text{control}$; **Parbegin** $c_1^1 = \sigma(c)$; $c_j^1 = \sigma(c_{j-1}^{l_{j-1}})$, where $j = 2 \ldots n$; $c_j^i = \sigma(c_j^{i-1})$, where $i = 2 \ldots l_j$ and $j = 1 \ldots n$; **If** (c_j) **then** substatements in level i of statement j, $\forall i = 1 \ldots l_j$, $j = 1 \ldots n$ **Parend**

Further details of NEL are omitted. In [6], the following is proved:

Theorem 1. *There is a compiler C that translates each program in the language NEL into a network. The constants (weights) that appear in the network are the same as those of the program, plus several rational small numbers. Furthermore, the size of the network is O(length) and its running time is O(execution mea-*

sure). Here, length is the static length of the program, i.e. the number of tokens listed in the source code, and the execution measure is its dynamic length, i.e., the number of atomic commands executed for a given input.

6 Rule Revision: Learning and Adapting

When the rules are written as a network, adapting the rules means adapting the weights of the associated network. For a fixed set of rules, configurations are created by the engine. Each configuration receives a score in the $[0, 1]$ range that describes the future performance of the plant starting with such a configuration. The score (or merit) is decided upon by the simulator NOXER of the nuclear plant. A weighted average score of the scores given to the various configurations created by the engine from a given ruleset is used as the evaluation of that ruleset. When this value is fed to the net, the net adapts its weights numerically. Inhibitions will be put on the weights which are related to the mandatory rules so that not to keep them unchanged during this learning process. The new rules (network) are expected to yield new configurations, and the learning process continues until reaching an acceptable performance. Our reinforcement learning implementation is peculiar in that **(i)** the scores do not measure the rules directly, but rather the configurations that the search algorithm yields using the rules; **(ii)** the implied input to our neural network (that describes the rules) is the simulator, and the net has no input-score pairs it learns, but rather an output for a fixed input; **(iii)** our scores are continous numbers in the continous [0,1] range rather than the more common binary score values, and the net is recurrent rather than the common feedback acyclic net.

References

1. Barto, A.G., Sutton, R.S., Watkins, C.J.: Learning and sequential decision making. In: M. Gabriel , J.W. Moore (eds.), Learning and computational neuroscience. Cambridge, MA: MIT Press 1991.
2. Galperin, A., Kimhi, Y., Nissan, E.: FUELCON: an expert system for assisting the practice and research of in-core fuel management and optimal design in nuclear engineering. Computers and Artificial Intelligence **12**, 4 (1993) 369-415.
3. Nissan, E., Siegelmann, H., Galperin, A., Kimhi, Y.: Upgrading automation for nuclear fuel in-core management: from the symbolic generation of configurations, to the neural adaptation of heuristics. (forthcoming).
4. Parks, G.T., Lewins, J.D.: In-core fuel management and optimization: the state of the art. Nuclear Europe Worldscan **12**, 3/4 (1992) 41.
5. Rothleder, B.M., Poetschhat, G.R., Faught, W.S., Eich, W.J.: The potential for expert system support in solving the Pressurized Water Reactor fuel shuffling problem. Nuclear Science and Engineering **100** (1988) 440 ff.
6. Siegelmann, H.T.: Foundations of recurrent neural networks. Ph.D. dissertation. New Brunswick, NJ: Rutgers University (1993).

Concept Hierarchies: a Restricted Form of Knowledge Derived From Regularities

Molly Troxel, Kim Swarm, Robert Zembowicz[†], and Jan M. Żytkow

Department of Computer Science, Wichita State University, Wichita, KS 67260-0083
[†] also JZB Information Systems, LLC, 9415 E. Harry, Suite #302, Wichita, KS 67207
{mltroxel, ksswarm, robert, zytkow}@wise.cs.twsu.edu

Abstract. In this paper we analyze relationships between different forms of knowledge that can be discovered in the same data matrix (database): contingency tables, equations, concept definitions, and concept hierarchies. We argue for the basic role of contingency tables and equations (law-like knowledge), and for the limitations of concept hierarchies. We show how a subset of contingency tables which approximate logical equivalence can be used to construct concept hierarchies: (1) each of those regularities leads to an element of the conceptual hierarchy, (2) the elements are merged to increase their empirical contents, and (3) hierarchy elements are combined into concept hierarchy. The possibility of different hierarchies leads to the question of choice between hierarchies, for which we provide our optimality criterion. We illustrate our algorithm by an application on the soybean database, and we show how our results go beyond the results obtained by the COBWEB approach to clustering.

1 Concepts and Regularities Discovered in Data

Relational tables in databases (data matrices in statistics, collections of examples in the machine learning research) have been used for a long time to seek different kinds of knowledge, for instance, rule-based concept definitions, taxonomies, and regularities in the form of equations and contingency tables. The relationships between these forms of knowledge, however, did not receive sufficient attention.

We argue that concepts and taxonomies include limited knowledge, compared to equations and contingency tables, which we jointly call law-like knowledge. Law-like knowledge is more expressive and easier to use. We demonstrate that an especially simple type of law-like knowledge discovered in data leads to useful boolean concepts and their hierarchies.

1.1 Concept Learning and Concept Discovery

In technical terms of logic, concepts are predicates which include free variables. They name objects, properties or patterns, but they are neither true nor false. Truth values can be assigned to statements, which use concepts and which have all variables bound by either explicit or implicit quantifiers. With the exception of tautologies, true and universally quantified statements are typically called laws or regularities.

A proven model of concept discovery comes from science. Concepts can be viewed as investments which produce payoff when they allow us to express regularities and laws. Better concepts can be recognized by analyzing regularities that they permit to express. Among an unlimited number of concepts that can be proposed, science uses very few, choosing them based on the generality, predictive power, accuracy, and number of laws in which they occur. In machine discovery we also use the same feedback [4,6,8] Our taxonomy formation algorithm presented in this paper, uses predictive strength, the scope, and the number of laws to guide concept formation.

Concept learning from examples can be viewed as a very limited search for regularities. Membership in the target class is described by an attribute, which indicates for each record, whether it belongs to that class or not. The learner seeks the best definition of the target class in terms of other attributes. Such a definition has a truth value. If true, it shares many features of regularities, for instance, it can be used to predict class membership. The target class is given by examples and counterexamples, and a learner seeks only a class definition. In contrast, a discoverer must find out on its own, which concepts are important. The path leads through discovery of regularities. A learner may not understand why a concept has value, a discoverer would, because the focus on regularities gives it a good foundation for the autonomous acceptance of concepts.

1.2 Clustering as Limited Discovery

Clustering is a step towards autonomy in concept learning. Here the task is more open, aimed at the autonomous creation of classes. Given a data matrix, clustering seeks to divide all records into classes and to find a description of each class. The concern for regularities in data has been notably absent in early clustering systems, and resultant taxonomies have had little scientific value. A new generation of clustering systems guides the clustering process by predictivity of clusters [1]. In addition to knowledge that is contained in descriptions of individual clusters, additional knowledge is implicit in cluster hierarchies when they are exhaustive and disjoint.

Knowledge included in a taxonomy falls into the category of monadic logic; membership criterion for each class is represented by a unary predicate, while empirical contents of each class and relations between classes are represented by equivalences and implications between such predicates. Knowledge represented by monadic predicates is very limited compared to the expressive power of equations and contingency tables, and to the expressive power of the first order logic.

Regularities of two or more dimensions are poorly represented by clusters and taxonomies. A regularity does not separate existing objects into classes, but it captures a pattern obeyed by all objects. In the clustering approach, many classes may be needed to represent predictivity of a simple pattern. For instance, a proportionality between x and y must be approximated by many clusters, rather than by a simple regularity $y = ax$. This applies also to functional relationships between non-numerical attributes, when these attributes have many values. When many regularities occur, their pieces are captured locally, in different combinations, by clusters. The search for law-like regularities leads to additional knowledge and to knowledge expressed in a more concise, useful form.

2 From Regularities to Concepts with Empirical Contents

Elsewhere [11] we argue that equations and contingency tables are two basic forms of regularities. The generic form of regularity is "Pattern P holds in domain D". A pattern that holds in a domain distinguishes events that are possible in that domain from events that are impossible. Contingency tables distinguish statistically probable combinations of events from those improbable. The majority of patterns do not imply new concepts. Take an equation in the form $y = f(x)$. It does not "naturally" lead to conceptual distinctions among the values of x or y, because all values of both variables are treated uniformly. The majority of contingency tables do not lead to "natural" concepts, either. A subcategory of contingency tables, however, gives strong reasons for concept formation. Furthermore, when a number of contingency tables in that category is inferred from data, and if these tables share some common attributes, we can use these regularities to form a concept hierarchy.

2.1 Concepts Inferred from Contingency Tables

Consider the table depicted in Figure 1, for boolean attributes A1 and A2. Nonzero numbers of occurrences of particular value combinations are indicated by n1 and n2. Zeros in the cells indicate that the corresponding combinations of values do not occur in data from which the table has been generated. For instance, no objects are both A1 and non-A2. From the zero values we can infer inductively, with the significance that increases as we consider more data, that these value combinations do not occur in the population represented by data.

A1	0	n1
¬A1	n2	0
	¬A2	A2

The regularity expressed in this table is equivalence:
For all x, (A1(x) if and only if A2(x))
2 classes can be defined: (1) A1 and A2, (2) non-A1 and non-A2

Fig. 1. Contingency table that leads to concepts of empirical contents.

The table motivates the partition of all data into two classes: (1) objects which are both A1 and A2, and (2) objects which are neither A1 nor A2. Each class has empirical contents, because we can determine class membership by the value of one attribute, and then predict the value of the other attribute.

The interpretation of zeros, illustrated in Figure 1 can be generalized to zeros that occur in tables of any size, but for large tables the inferred concepts and their properties may be too many and too weak.

2.2 Approximate Inference

In real databases, rarely we see functional regularities without exceptions. Instead of cells with zero counts, we can expect cells with numbers small compared to those in other cells. In our method we want to tolerate limited exceptions.

Rather than directly compare the numbers in different cells to determine whether a table approximates equivalence, we use Cramer's V, set at a threshold close to 1.0. The Cramer's V coefficient is based on χ^2, which measures the distance between tables of actual and expected counts. For a given $M_{row} \times M_{col}$ contingency table, Cramer's V is defined as

$$V = \sqrt{\chi^2/(N \min(M_{row} - 1, M_{col} - 1))},$$

where N is the number of records. Cramer's V is a measure of the predictive power of a regularity. The regularity between x and y has a larger predictive power if for a given value of x the value of y can be predicted more uniquely. The strongest, unique predictions are possible when for each value of one attribute there is exactly one corresponding value of the other attribute. For ideal correlation, χ^2 is equal to $N \min(M_{row} - 1, M_{col} - 1)$, so Cramer's $V = 1$. On the other extreme, when the actual distribution is equal to expected, then $\chi^2 = 0$ and $V = 0$. Cramer's coefficient V does not depend on the size of the contingency table nor on the number of records. Thus it can be used as a homogeneous measure on regularities found in different subsets and for different combinations of attributes. In addition to Cramer's V, we use significance to qualify regularities for further analysis.

3 From Regularities to Taxonomies

As an example that illustrates our method for taxonomy formation we selected the small soybean database of 47 records and 35 attributes, because it has been extensively studied [1,5,9].

We used the 49er system [11] to discover two-dimensional regularities in soybean data, for all combinations of attributes in all data and in a large number of subsets. Systems such as EXPLORA [2,3] could be also applied to derive the relevant contingency tables.

We set 49er parameters so the system seeks only the regularities in the form of contingency tables, for which the Cramer's V values ≥ 0.90. An example of such a finding, reported in Table 1 is a regularity between the attributes stem-cankers and fruiting-bodies, with the Cramer's V rating of 1. Note that the value of fruiting-bodies (0 or 1) can be uniquely predicted for each value of stem-cankers. 49er found many such regularities, which strongly suggests that the database is conducive to taxonomy formation.

Starting from the regularities found by 49er, the following algorithm is used to build the hierarchy of concepts:

```
Main Algorithm: Build class hierarchy
    Create hierarchy elements
    Merge similar hierarchy elements
    Build hierarchy tree from merged elements
```

In the following subsections we describe details of our algorithm.

FRUITING-BODIES

FRUITING-BODIES	STEM-CANKERS 0	1	2	3
1	0	0	0	10
0	10	18	9	0

STEM-CANKERS

Range: All records (47)
Cramer's V = 1.0
Chi-square = 47.0

Table 1. A regularity found in the small soybean dataset by 49er's search for regularities. The numbers in the cells represent the numbers of records with combinations of values indicated at the margins of the table.

3.1 Hierarchy Elements Generated from Equivalences

Each contingency table, for which Cramer's V meets the threshold, is used to build an elementary hierarchical unit:

```
Procedure: Create hierarchy elements
    for each regularity
        if regularity strength exceeds threshold
            Form hierarchy element from regularity
```

A hierarchy element is a simple tree, comprised of 3 classes: the root and two children (see Figure 2). The root is labeled with the description of the class of records, in which the regularity holds. Each child is labeled with the descriptors which hold for that child based on the considered regularity. An example of a descriptor is Stem-Cankers(0,1,2): "the values of Stem-Canker are 0, 1, or 2". The children classes are approximately disjoint and they exhaustively cover the range of the regularity.

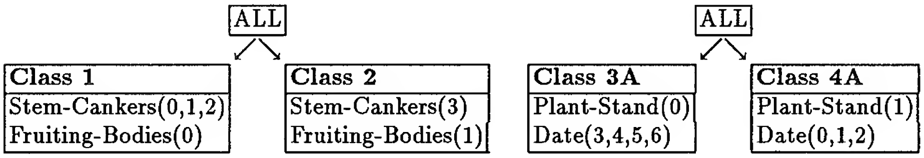


Fig. 2. Examples of two hierarchical elements built from the regularities for (i) Fruiting-Bodies and Stem-Cankers, and (ii) Plant-Stand and Date. Both regularities hold for all data, hence the root is ALL in both cases.

3.2 Merging the Hierarchy Elements

If the same class can be derived from different regularities, it will occur in different hierarchy elements and will be characterized by many descriptors. To identify different occurrences, after each hierarchy element is created, it is compared to all other elements over the same range of records, in search for common descriptors. If they were found in the same data set and have a common descriptor (the same attribute and equal value sets), the classes are identical (approximately identical, because of exceptions, as discussed above). The complementary classes must be

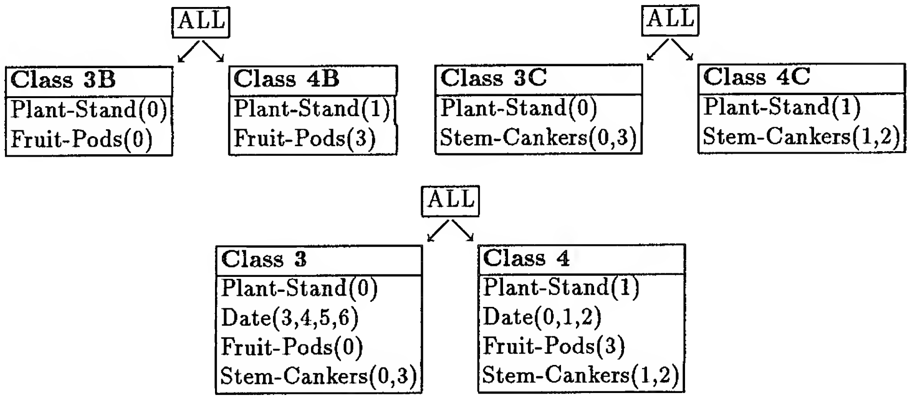


Fig. 3. Hierarchy elements formed from the regularities between (i) Plant-Stand and Fruit-Pods (upper left) and (ii) Plant-Stand and Stem-Cankers (upper right), share the same descriptor (Plant-Stand) as the hierarchy element found for the regularity between Plant-Stand and Date (right part of Figure 2). All three hierarchy elements are merged together (lower part).

also identical. Both hierarchy elements are collapsed into one and the descriptors for the corresponding children are merged (Figure 3).

The same algorithm applies recursively to regularities found in subsets of data. For a regularity in a subset described by condition C, the root of the hierarchy element is labeled by C (example in Figure 4).

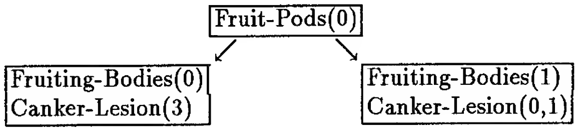


Fig. 4. A hierarchy element over a subrange of the soybean data defined by the descriptor Fruit-Pods(0).

The search through the soybean database produced 15 regularities holding for all data, initially leading to 30 classes and to 8 classes after merging.

3.3 Definitional and Inferred Properties

The children classes in each hierarchy element contain a number of descriptors. In our soybean example, after all possible merges of hierarchy elements, Classes 3 and 4 contain four descriptors (Figure 4), while Classes 5 and 6 contain six other descriptors each. Each of those descriptors can be used to define membership in a given class. We call such a descriptor definitional. If a definitional descriptor D is used to define membership in C, all other descriptors can be deduced, leading to predictions of property values for members of C.

Since we allow exceptions, a conjunction of two definitional descriptors may offer a more reliable recognition, but will reduce the number of deduced descriptors.

The choice available among definitional descriptors of concept C offers flexibility in selection of the recognition procedure for C. Depending on the observable concepts available, one or another definitional descriptor can be used. Since alternative recognition procedures can be applied, missing values pose a lesser problem in our approach in contrast to many machine learning systems.

Each definitional descriptor D for a class C in a hierarchy element H is sufficient to determine whether a given record belongs to C only within the range of H. To obtain a complete definition of C, we must use D in conjunction with the definition of the range of the hierarchy element, which can be assembled from descriptors definitional for each node on the path from the root. Being able to make a choice of a definitional descriptor at each level, we can assemble the complete definition in a very flexible way.

If every object in class C satisfies descriptor D, but not the other way, D is a property of objects in C, but it does not define membership in C. We will call such a descriptor D an inferred property. Regularities in the form of contingency tables of the "implication" type (see section 2.1) can be used to obtain inferred properties. Also all descriptors at the higher levels of the hierarchy than C are inferred descriptors for C.

3.4 Empirical Contents of a Concept

Each concept can be characterized by its extent and intent. The extent is the set of all objects which are instances of the concept. The intent is the set of property values (represented by descriptors) possessed by all objects in the extent.

We postulate that the empirical contents, which can be also called predictive contents of a concept, is proportional to the number of descriptors which can be deduced about a single object in C, after the membership has been determined. We can further postulate that the empirical contents is also proportional to the cardinality of the extent of the concept, because for each object in the extent, once it is recognized as a member of C by testing one definitional property, other descriptors can be deduced.

Empirical contents measures the significance of a concept. It can be used to decide on concept acceptance, and to make choices between concepts. By totaling the empirical contents of individual concepts we can also define the empirical contents of the whole taxonomy.

3.5 Taxonomy Formation

The following procedures transform the list of elementary hierarchy units into a multi-level taxonomy, which is exhaustive and (approximately) disjoint at each level:

```
Procedure: Build hierarchy tree from elements
    Sort hierarchy elements according to the decreasing number of descriptors
    hierarchy tree ← empty
    for each hierarchy element
        Add hierarchy element to hierarchy tree
```

```
Procedure: Add hierarchy element to hierarchy tree
  for each leaf in the hierarchy tree
    Attach children in the hierarchy element to the leaf
    Remove children incompatible with the path to the root
    if there is only one child left
      Merge this child with its parent
```

For our soybean example, this algorithms puts classes 5 and 6 at the uppermost level of the hierarchy tree, then classes 3 and 4, etc. (Table 2). We position the classes with greater number of descriptors above those with less descriptors, to minimize the number of times each descriptor occurs in the taxonomy.

Some nodes in the nascent hierarchy can be empty. This possibility is examined as soon as new node is added, by computing the intersection of the value sets for each common attributes from the new node upward to the root of the taxonomy. If for a common attribute this intersection is empty, no objects in the dataset can possibly belong to the new node. This node is then eliminated. For example, Class 6 contains the attribute Stem-Cankers with the value range (0), and under it Class 4 contains the same attribute with the value range (1,2). No records in the data can be in both these classes simultaneously, therefore the class 4 under Class 6 is eliminated (shown in *italic* in Table 2).

In Table 2 we have shown only the use of regularities for all data, with one exception, marked with (*). The values of Canker-Lesion (CL-3 and CL-0,1), shown in two locations under Class 3, come from a regularity found in the subset of data defined by Fruit-Pods=0. That regularity, depicted in Figure 4, links Canker-Lesion to Fruiting-Bodies. Since the same values of Fruiting-Bodies define Class 1 and Class 2, the corresponding values of Canker-Lesion become inferred descriptors in the subclasses of Class 1 and Class 2 within Class 3.

When after elimination of empty classes only one child class remains under a parent class, the descriptors of this class are added to the descriptor list of the parent class, expanding the intent of the parent node. The descriptors acquired from the lower class become inferred properties in the parent class, because all objects in the parent class also belong to the remaining lower class. We see in Table 2 that Class 4 is eliminated under Class 6, therefore any object in Class 6 is included in Class 3. However, Class 3 also contains objects that belong in Class 5, so the merged properties from Class 3 to Class 6 cannot be definitional, but merely inferred. The descriptors of Class 7 and Class 1 under Class 6 are also included as inferred descriptors for Class 6. Table 3 presents all definitional and inferred descriptors of Class 6, reached by our algorithm.

After all the pruning, we are left with the four nodes at the bottom level in Table 2. These nodes, along with three internal nodes (cf Figure 5) form the taxonomy. We can hypothesize that this taxonomy describes the natural divisions of the soybean database's diseases. It turns out that the extents of the four leaf concepts in our taxonomy are equal to the four diseases, listed under each leaf in Table 2 as D1 through D4. We can hypothesize that the internal nodes correspond to natural classes of diseases.

Class 5	Class 6
Internal Discolor: 0	Interal Discolor: 2
Precipitation: 1,2	Precipitation: 0
Sclerotia: 0	Sclerotia: 1
Area Damaged: 0,1	Area Damaged: 2,3
Stem Cankers: 1,2,3	Stem Cankers: 0
Canker Lesions: 0,1,2	Canker Lesions: 3

Class 3	Class 4	Class 3	*Class 4*
Stem Cankers: 0,3	Stem Cankers: 1,2	Stem Cankers: 0,3	*Stem Cankers: 1,2*
Plant Stand: 0	Plant Stand: 1	Plant Stand: 0	*Plant Stand: 1*
Date: 3,4,5,6	Date: 0,1,2	Date: 3,4,5,6	*Date: 0,1,2*
Fruit Pods: 0	Fruit Pods: 3	Fruit Pods: 0	*Fruit Pods: 3*

Class 1	Class 2	Class 1	*Class 2*	Class 1	*Class 2*
SC: 0,1,2	SC: 3	SC: 0,1,2	*SC: 3*	SC: 0,1,2	*SC: 3*
FB: 0	FB: 1	FB: 0	*FB: 1*	FB: 0	*FB: 1*
CL: 3(*)	CL: 0,1(*)			CL: 3(*)	*CL: 0,1(*)*

C7	*C8*	C7	C8	C7	*C8*
CL: 0,1,3 R: 0	*CL:* *2* *R: 1*	CL: 0,1,3 R: 0	CL: 2 R: 1	CL: 0,1,3 R: 0	*CL:* *2* *R: 1*
D1		D3	D4	D2	

Table 2. The taxonomy generation process, depicted from the top (Classes 5 and 6) till the bottom (Classes 7 and 8). Empty classes are shown in *italic* font. Abbreviations: FB = Fruiting-Bodies; R = Roots; SC = Stem-Cankers; CL = Canker-Lesions

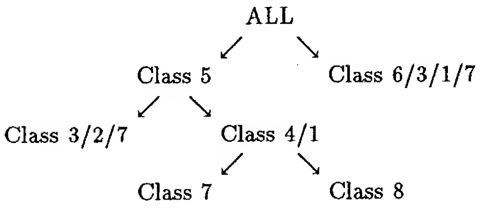


Fig. 5. The finished taxonomy, after pruning empty nodes and merging single children.

Our algorithm places concepts with higher empirical contents at the top of the taxonomy. This minimizes the number of descriptors which must be stored in the taxonomy. Our taxonomy, shown in Table 2 requires 33 descriptors, while a taxonomy which puts classes with the fewest attributes at the top (classes 7&8, followed by classes 1&2, 3&4, and 5&6 at the bottom, requires 44 descriptors.

4 Comparisons

In our method, concept taxonomies are secondary, as they can be formed from regularities in the form of special contingency tables. Previous conceptual clustering systems use various methods to find cluster taxonomies, but as they do not consider law-like regularities, their approximation methods may miss important knowledge useful in taxonomy formation.

Linear Clustering, introduced in [7], uses regularities, but in another way. When several linear patterns are detected in the same domain, clusters are formed that capture the subdomains in which each linear pattern is unique.

Wu, et al.[10] combine Bayesian and expert knowledge of a domain in the form of domain heuristic measures to guide classification heuristics. In distinction, our approach requires no prior domain knowledge to determine meaningful classes in the data.

	OUR SYSTEM		COBWEB		
Attribute	Value	Role	Value	[P(value|D2), P(D2|value)]	Role
Precipitation	0	Definitional	0	[1.0 , 1.0]	Definitional
Temperature	1,2	Inferred	2	[0.6 , 1.0]	Partial Def.
Stem Cankers	0	Definitional	0	[1.0 , 1.0]	Definitional
Fruit Pods	0	Inferred	0	[1.0 , 0.5]	Weak
Canker Lesion	3	Definitional	3	[1.0 , 1.0]	Definitional
External Decay	0	Inferred	0	[1.0 , 0.48]	Weak
Internal Discolor	2	Definitional	2	[1.0 , 1.0]	Definitional
Sclerotia	1	Definitional	1	[1.0 , 1.0]	Definitional
Area Damaged	2,3	Definitional			Not included
Plant Stand	0	Inferred			Not included
Date	3,4,5,6	Inferred			Not included
Fruiting Body	0	Inferred			Not included
Roots	0	Inferred			Not included

Table 3. The comparison of our results with the COBWEB system on one concept (Disease 2 - Charcoal Rot). Our method finds all 8 descriptors captured by COBWEB, as well as 5 additional descriptor, including one definitional descriptor, Area Damaged(2,3). The descriptors for Temperature and External Decay were derived in a subrange of records, and were not included in Table 2 due to space restrictions. P(A|B) means the probability of being in A for the objects in B.

In Table 3 we compare the results of our algorithm to the available results obtained by COBWEB [1]. The comparison concerns Class 6, which corresponds to Disease 2, called Charcoal Rot. Notice a substantial increase of empirical contents reached in our approach, which shows in the number of additional predictions possible for five attributes of the inferred type, not included by COBWEB.

COBWEB adds objects incrementally to the conceptual hierarchy, change operators are used to merge, split and delete nodes from the hierarchy. Our algorithm is not incremental. It forms a hierarchy from regularities already discovered by a database mining system, so that the change operators are unnecessary.

5 Summary

We have presented a theory and an algorithm for conceptual hierarchy formation from law-like knowledge in the form of contingency tables. We used the soybean database as an example. It turned out that four diseases hidden in the soybean data coincide with the four leaves in taxonomies generated by our algorithm.

We argued that knowledge contained in concepts is secondary to knowledge contained in regularities. Autonomous discoverer will see reasons to claim empirical contents of some concepts when coexisting properties are revealed by regularities. We argued that better concepts are those with higher empirical contents and therefore a knowledge discovery system should notice that contents, and use it in concept formation.

In contrast to fixed rules which define concepts in many machine learning systems, our approach offers the choice among definitional descriptors in selecting a recognition procedure. Depending on the observable concepts available in a given situation, alternative definitional descriptors can be used, so that missing values do not pose a problem in our approach in contrast to many machine learning systems.

References

1. Fisher, D.H. 1987 . Knowledge Acquisition Via Incremental Conceptual Clustering *Machine Learning 2* 139-172.
2. Hoschka, P. & Klösgen, W. 1991. A Support System for Interpreting Statistical Data, in: Piatetsky-Shapiro G. & Frawley W. eds *Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press, 325-345.
3. Klösgen, W. (1992). Patterns for Knowledge Discovery in Databases in: ed *Proceedings of the ML-92 Workshop on Machine Discovery (MD-92)*, Aberdeen, UK. July 4, p.1-10.
4. Langley, P., Simon, H. A., Bradshaw, G. L. & Żytkow, J. M. 1987. *Scientific discovery: Computational explorations of the creative processes.* Cambridge, MA: MIT Press.
5. Michalski, R.S. & Chilausky, R.L. 1980. Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis, *Int.J. of Policy Analysis and Info. Systems*, 4, 125-161.
6. Nordhausen, B. & Langley, P. 1993. An Integrated Framework for Empirical Discovery, *Machine Learning*, 12, 17-47.
7. Piatetsky-Shapiro, G. & Matheus, C. 1991. Knowledge Discovery Workbench: An Exploratory Environment for Discovery in Business Databases, in Piatetsky-Shapiro ed. *Proc.of AAAI-91 Knowledge Discovery in Databases Workshop*, 11-24.
8. Shen, W. 1993. Discovery as Autonomous Learning from the Environment. *Machine Learning*, 12, 143-165.
9. Stepp, R.E. 1984. Conjunctive Conceptual Clustering: A methodology and experimentation, Ph.D. dissert., Dept. of Computer Science, Univ. of Illinois, Urbana.
10. Wu, Q., Suetens, P. & Oosterlinck, A. 1991. Integration of Heuristic and Bayesian Approaches in a Pattern-Classification System, in Piatetsky-Shapiro G. & Frawley W. eds. *Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press, 249-260.
11. Żytkow, J., & Zembowicz, R., (1993) Database Exploration in Search of Regularities, *Journal of Intelligent Information Systems*, 2, p.39-81.

A Data-Driven Approach to Feature Construction

Jianping Zhang and Hsueh-Hsiang Lu

Department of Computer Science
Utah State University
Logan, Utah 84322-4205 USA

Abstract. This paper presents a general scheme for feature construction and its application to decision trees. In this scheme, a higher level attribute is constructed from two lower level ones under the guidance of the distributions of the examples from different classes. It can be used with different selective induction algorithms such as decision tree learning, rule learning and instance-based learning. A simple prototype has been implemented and integrated into a decision tree learning system. The experimental results empirically show that our approach outperforms the standard decision tree learning algorithm on three domains tested.

1 Introduction

The task of concept learning is to find a concept description from a set of training examples. In attribute-based concept learning, each training example is described by a vector of values of predefined attributes with its class membership. The concept description generated by a learning system should be able to predict the class membership of new examples of the concept. Many existing concept learning systems such as CN2 [1] and ID3 [7] perform selective induction, that is, the concept description language is the same as the language used to describe examples. These selective induction algorithms work fine when training examples are described by directly relevant attributes. However, when attributes are not directly relevant to the target concept, selective induction algorithms fail to perform well [8]. When problems are well understood, human experts can easily generate all relevant attributes so that selective induction algorithms can be applied. However, many problems in real world applications are poorly understood and it can be difficult and time consuming to produce relevant attributes. To make learning algorithms applicable to these poorly understood problems, constructive induction or feature construction [5] needs to be incorporated.

Constructive induction has been widely studied in recent years. Rendell and Seshu [8] proposed a general framework for constructive induction. Drastal et al. [2] reported a supervised rule learning system MIRO which performs constructive induction under the guidance of a domain theory. Matheus [4], Pagallo [6] and Yang et al. [12] introduced approaches to constructive induction on decision trees. Other work on constructive induction includes [9, 10,11, 13].

In this paper we present a general scheme for feature construction and its application to decision trees. In our scheme, the input is two lower level attributes, the output is a higher level attribute constructed from the two lower level ones. Feature construction in this scheme is guided by the distributions of examples from different classes. Constructed attributes may be binary or multi-valued, and primitive attributes can be multi-valued or continuous. Generalizations may be performed in feature construction. Our scheme constructs an attribute by partitioning the plane formed by two input attributes into several regions based on the distributions of examples from different classes. Each of the regions forms a value of the constructed attribute. The boundary of a region does not have to be parallel or orthogonal to both attribute axes. This scheme can be used with different selective induction algorithms such as decision tree learning, rule learning and instance-based learning.

A simple version of the scheme has been implemented and applied to a decision learning algorithm. The implemented system has been tested in several artificial domains. The results of these preliminary experiments demonstrate the scheme's ability to improve the learning performance through constructive induction.

2 A Feature Construction Scheme

This section presents a novel approach to feature construction. Before presenting the approach, we first introduce a framework for concept learning with constructive induction. Our approach fits the framework and performs the task of constructive induction.

2.1 A Framework for Concept Learning with Constructive Induction

In this subsection, we present a framework for concept learning with constructive induction. This framework is similar to the frameworks reported in [8, 4, 6]. In this framework, selective induction is first applied to generate a concept description. If the quality of the generated concept description is not high enough, constructive induction is applied to create new features. Then, selective induction is applied again. This process is repeated until the quality of the concept description is satisfied or no new feature can be constructed. The framework is described as follows:

```
Initialize the set ATT to a set of primitive attributes for expressing training
instances.
Repeat
  If ATT contains too many attributes,
     use some attribute quality measure to remove some attributes from ATT.
  Express training instances using attributes in ATT.
  Apply a standard selective induction algorithm to learn a concept description.
  If the quality of generated concept description is improved,
     construct new attributes and add them into ATT
until the quality of the generated concept description is no longer improved or no
     new attribute can be generated.
```

As described in [8], a concept description quality measure evaluates some criterion such as the accuracy or conciseness of a concept description and an attribute quality measure evaluates the relevance of an attribute to the target concept.

2.2 Constructing Features

In this subsection, we present an approach to combine two lower level attributes to form a new higher level attribute. More complex attributes may be created through the iterative process in the framework introduced in section 2.1. For example, a combination of three attributes may be formed through two iterations. For simplicity, we assume all primitive attributes have discrete values.

Given two attributes A1 and A2 with values (0, 1, ..., n-1) and (0, 1, ..., m-1) respectively, the two dimensional space formed by A1 and A2 includes n * m points (0 0), (0 1), ..., (n-1, m-2), (n-1, m-1). Creation of a new attribute from A1 and A2 can be viewed as a partition of the n * m points. For example, the conjunctive feature of two binary attributes is a binary partition of the four points (0 0), (0 1), (1 0) and (1 1), and the two subsets of the partition are {(1 1)} and {(0 0), (0 1), (1 0)}. Each subset of a partition is a value of the constructed attribute. Therefore, the problem of constructing a new attribute becomes the determination of a partition in a two dimensional space. Construction of a binary attribute is a binary partition. A multi-value attribute can be constructed by partitioning the two dimensional space into more than two subsets.

In the past, some work has been done in grouping values of multi-valued (or continuous) attributes [3,7]. Actually, grouping values of an attribute is a simple form of constructive induction, because it constructs a new attribute which has different values from the old attribute. Our approach to feature construction can be viewed as an extension of these approaches to grouping values.

Our approach can be divided into two steps. The first step constructs a new attribute by simply combining two lower level attributes with m and n values respectively. The new attribute has m * n values, namely, each individual point of the two dimensional space forms a value of the new attribute. An attribute constructed in this way often has too many values, especially when a constructed attribute is a combination of many primitive attributes. The number of values of a constructed attribute increases exponentially with the number of primitive attributes involved in the constructed attribute. For example, an attribute constructed from ten binary primitive attributes have 2^{10} different values, each of which is a point in the ten dimensional space. Constructive induction is a way of detecting hidden relevant patterns, but constructive induction performed above does not generate any hidden relevant patterns. Moreover, the attribute constructed in this way does not generalize beyond its input data.

The second step of our approach is to reduce the values of a constructed attribute by grouping values of the attribute constructed in the first step. Approaches reported in [3,7] can be used to group values in our approach. In this paper, we propose a novel approach to partitioning points on a two dimensional space. We first introduce the approach to binary partitions. A binary partition results in a binary attribute. Assume that the two subsets of a binary partition are S1 and S2. A point (value) p belongs to S1 if $\frac{P_p}{Pos} > \frac{N_p}{Neg}$, where Pos and Neg are the total number of positive and negative examples respectively; P_p and N_p are the number of positive and negative examples on the point p. S2 contains all points on which $\frac{P_p}{Pos} \leq \frac{N_p}{Neg}$. Here, a point p is a value of the attribute constructed in the first step. S1 contains all points which favor the target concept, while S2 covers all points which are against the target concept. The rationale for this method is that the points which favor the target concept and the points which are against the target concept should not be mixed.

The binary partition method discussed above can be extended to a method which partitions all points into more than two subsets. For each point p, we first compute

$\frac{Pp}{Pos} - \frac{Np}{Neg}$. Then, all points are sorted according to the magnitude of the difference. The next step is to find several cut points on these sorted points. The binary partition method discussed above is a special case of this method and the only cut point for a binary partition is 0. To find cut points is to find big gaps on the sorted points. Each big gap implies a cut point. All points between two consecutive cut points form one subset, thus a value of the constructed attribute. If no gap exists, the binary partition method can be applied.

A different partition of the two dimensional space constructs a different attribute. For example, we want to construct a new attribute from two binary attributes, A1 and A2 with values 0 and 1. The partition S1 = {(1 1)}, S2 = {(0 0), (0 1), (1 0)} constructs a conjunctive feature of A1 and A2, the partition S1 = {(1 1), (1 0), (0 1)] S2= {(0 0)} generates a disjunctive feature of A1 and A2, while the partition S1 = {(1 1), (0 0)}, S2 = {(1 0), (0 1)} creates an exclusive-or of A1 and A2. Under our approach, the construction of disjunctive, conjunctive, or exclusive-or attributes is unified and depends on the distributions of positive and negative examples.

Many of the attributes constructed by combining two attributes are not relevant to the target concept. Creation of many irrelevant attributes may confuse a selective induction algorithm and increase learning difficulty. Therefore, it is important to reduce the number of irrelevant attribute constructed. In our approach, a new attribute is created only if it is more relevant than the two original attributes. Each time an attribute is created, its relevance is computed using the information theoretic measure. If its relevance is higher than those of the two original attributes, it is output as a new attribute. Otherwise, no new attribute is created. A more conservative method is that a new attribute is created only if it is significantly better than the two original attributes. Another way to limit the number of irrelevant attributes generated is when this approach is integrated into a selective induction algorithm, some heuristics similar to those used in [6] can be used to propose attribute pairs which have big chances to produce highly relevant attributes.

The method proposed above works only if there are enough examples observed on every point. Therefore, no generalization is performed. Generalization is important in constructive induction [4], when training set is sparse. This problem can be approached in the following way. The partition is only performed on those points on which significant amount of examples are observed. Each subset obtained above is generalized by incorporating the points on which only few examples are observed. The generalization of a subset is another concept learning problem. The instance space of the learning problem is the two dimensional space of the two input attributes. The training set consists of all points in the two dimensional space with significant amount of observed examples, and concepts are all subsets generated by the partition. The goal is to learn a description for each subset so that the points without many observed examples can be classified to one of the existing subsets. Simply, we can apply an instance-based learning approach to perform this task. Because of the small instance space (two dimensional space), the learning problem is very simple. Therefore, we can design an inductive learning algorithm with a powerful description language. This powerful description language should be able to represent a region with a boundary that is not orthogonal to both axes.

Let us consider a simple example. Assume that there are two attributes: Length and Width which represent the length and the width of a rectangle respectively, and the target concept is the concept of square. These two attributes are not directly relevant to the target concept. To learn the target concept, a new attribute is ought to be constructed. First, all points in the two dimensional space of Length and Width with observed examples are partitioned into two subsets, one subset S1 contains all points on which only positive examples are observed, and the other subset S2 contains all points on which only negative examples are observed. There is no point on which both positive and negative examples are observed. Points in S1 and points in S2 are the positive examples and negative examples of the new learning problem. Because all points in S1 are on the line Length = Width, so the description generated from points in S1 is Length = Width. Correspondingly, the description generated from all points in S2 is Length ≠ Width. The constructed attribute has two values, one is Length = Width and the other is Length ≠ Width.

Another solution for sparse training sets is a pattern directed approach. In a pattern directed approach, some predefined patterns, such as or, and, exclusive-or, x = y, can be stored in a pattern base. When a training set is sparse so that it becomes hard to detect relevant patterns, a partition on the points with enough observed examples is generated first. Then, the pattern base is searched for a predefined pattern which is the most consistent with the partition generated. A predefined pattern is similar to a constructive operator defined in [4]. The distribution of examples can help us decide a constructive operator. The pattern directed approach is a simple form of knowledge-directed approach too. For each different application, domain-dependent patterns can be added to the pattern base.

The approach reported in this section can be extended to handle continuous attributes. The two dimensional space formed by two continuous attributes is first broken into small rectangular areas, then our approach can be applied. Our approach can be also extended to construct attributes in a multi-class learning problem. We do not discuss the extension in this paper.

3. A Prototype: FEACON and Decision Tree Learning

We have implemented a simple prototype of the approach to feature construction introduced in section 2. In this prototype, primitive attributes are limited to nominal attributes and can be multi-valued. Constructed attributes have either two or three values. This prototype has been integrated into a decision tree learning system and tested on several artificial domains. This section discusses the prototype and its application to a decision tree learning. Section 4 reports the experimental results.

3.1 FEACON

Our prototype is named as FEACON. The input of FEACON is two nominal attributes and a set of positive and negative examples of the target concept. The output is an attribute which is a combination of the two input attributes and has higher relevance than both input attributes. The attribute constructed may have two or three values. If none of the combinations of the input attributes has higher relevance, no new attribute is output.

FEACON has two modes. Under the first mode, FEACON constructs a binary attribute. To construct a binary attribute, FEACON partitions the two dimensional space constituted by the input attributes into two subsets, one subset contains all points on which $\frac{P_p}{Pos} > \frac{N_p}{Neg}$ and the other subset contains points on which $\frac{P_p}{Pos} <= \frac{N_p}{Neg}$, where Pos and Neg are the total number of positive and negative examples, respectively; P_p and N_p are the number of positive and negative examples on the point p, respectively. Each of the two subsets is a value of the constructed attribute.

The second mode allows FEACON to construct an attribute with two or three values. To construct an attribute with three values, FEACON then tries to partition all points in the two dimensional space of the two input attributes into three subsets. The first subset contains all points on which $\frac{P_p}{Pos} - \frac{N_p}{Neg} > e$. The second subset contains all points on which $\frac{N_p}{Neg} - \frac{P_p}{Pos} > e$. The third subset includes all points on which $\left| \frac{P_p}{Pos} - \frac{N_p}{Neg} \right| \leq e$. e is a user defined parameter which takes a value between 0 and 1. The first subset favors the target concept, the second subset is against the target, while the third subset is irrelevant to the target concept. Each subset corresponds to a value of the constructed attribute. If one of the three subsets is empty, a binary attribute is created.

After an attribute is constructed, its relevance is evaluated using a measure similar to the one used in [7]. If its relevance is higher than both of the input attributes, the constructed attribute is the output of FEACON. Otherwise, no attribute is constructed.

3.2 Application of FEACON to Decision Tree Learning

FEACON has been integrated into a decision tree learning algorithm. The integrated learning system, called FEACONDT, fits the framework presented in section 2.1. The decision tree learning algorithm performs selective induction, while FEACON performs constructive induction. FEACONDT can be described as follows:

```
Initialize the set ATT to a set of primitive attributes for expressing training
        instances.
Repeat
    If ATT contains too many attributes,
        remove some attributes with the lowest relevances from ATT.
    Express training instances using attributes in ATT.
    Apply a decision tree learning algorithm to generate a decision tree.
    If the tree has fewer nodes than the previous tree,
        for each pair of the attributes at the two nodes nearest to a leaf,
            apply FEACON to construct a new attribute and add it into ATT
until the tree does not have fewer nodes than the previous tree or no new
        attributes can be generated.
```

The decision learning algorithm is similar to ID3, but has no pruning. For small problems (problems with a few attributes), FEACON can be applied to each pair of attributes. However, when a problem has a large number of attributes, this method tremendously increases the complexity of the system. To overcome the problem, we use a heuristic to reduce the number of pairs to be tried by FEACON. The heuristic is similar to the one used in FRINGE [6]. Namely, FEACON is only applied to the two attributes which are at the two nodes nearest to a leaf.

4 Experiments

We conducted some preliminary experiments with the FEACONDT system on six artificial domains: 4-term 3DNF, DNF with multivalued attributes, 4-term 3CNF, CNF with multivalued attributes, parity4 and parity5 functions. The performance was evaluated on two aspects: classification accuracy and conciseness of a concept description. Classification accuracy was measured as the percentage of correct classifications made by a concept description on a set of randomly selected test instances. Conciseness was measured as the number of nodes in a decision tree. The results obtained by FEACONDT were compared with the results obtained by the standard decision tree learning algorithm and FRINGE. Table 1 summaries the experimental results. All results in Table 1 are the average of ten trials. In our experiments, all training and test sets are randomly selected. The number of test examples is 1000 in all experiments. All three algorithms: FEACONDT, standard decision learning algorithm, and FRINGE do not perform pruning.

The 4-term 3DNF was randomly selected from those used in [4]. The 4-term 3DNF function has sixteen attributes. The results show a significant improvement of FEACONDT on both accuracy and conciseness over the standard decision tree learning. FEACONDT did not perform as good as FRINGE on small training set sizes. This result is expected and can be explained below. First, FEACONDT is a statistical approach so its effectiveness is based on a large amount of data available. Second, feature construction in FEACONDT is more general than FRINGE which is especially effective in learning small DNF [6]. The generality reduces the effectiveness on some specific problems.

DNF with multi-valued attributes: MDNF was to test how FEACONDT performs on multi-valued attributes. In this problem, each attribute has 6 values, 0 to 5. The DNF function is the same as the above 4-term 3DNF, but the value of a conjunct in a 4-term 3DNF was replaced with a disjunction of several values randomly selected from 0 to 5. The target concept has four disjuncts, each of which has three conjuncts, each of which, in turn, has two or more internal disjuncts. If the sixteen multi-valued attributes are converted to binary attributes, the target concept would have about 100 disjuncts. Therefore, this is a very hard learning problem. FEACONDT performed significantly better than both the standard decision tree algorithm and FRINGE in terms of both classification accuracy improvement and tree conciseness, especially for large train set sizes. FRINGE only slightly outperformed over the standard decision tree, this is due to the inability of FRINGE to feature construction of multivalued attributes.

A CNF is a conjunctive of disjuncts. A 4-term 3CNF has four conjuncts each of which is a disjunct of three literals. As indicated in [6], a CNF is a more difficult concept to learn than a DNF for a decision tree learning algorithm. It is interesting to see that the results achieved by FEACONDT in 4-term 3CNF is similar to those in 4-term 3DNF. FRINGE did not improve much over the standard decision tree, this is because FRINGE can only construct conjunctive new attributes. Dual-FRINGE [6] was designed to overcome the problem.

Similar to MDNF, attributes in CNF with multivalued attribute (MCNF) have 6 values, 0 to 5. The CNF function is the same as the 4-term 3CNF, but each literal was replaced with the disjunction of several values randomly selected from 0 to 5. FEACONDT again significantly improved the performance over both the standard decision tree and FRINGE. FRINGE achieved about same performance as the standard decision tree. The results of FEACONDT was similar to and slightly better than the result in MDNF.

The parity functions are parity four PAR4 and parity five PAR5. Additional eight and eleven irrelevant attributes were added to PAR4 and PAR5, respectively. As shown in table 1, FEACONDT performed better than FRINGE on both parity functions when training set sizes were 200, 400 and 800. FRINGE performed better for training set size 1600. The reason for this result is not very clear and needs to be explored in the future.

Table 1: Experimental Results

Domain	Training	ave. % error			ave. tree size		
	Set Size	DTree	FRINGE	FEACONDT	DTree	FRINGE	FEACONDT
DNF	200	19.9	0.0	8.5	80.2	9.0	36.2
	400	14.0	0.0	0.1	133.6	9.0	11. 0
	800	6.8	0.0	0.0	173.4	9.0	8.2
	1600	3.0	0.0	0.0	207.6	9.0	4.6
MDNF	200	46.7	46.7	38.6	135.9	135.9	67.4
	400	40.1	37.7	31.5	243.9	230.5	131.8
	800	35.8	32.2	17.2	463.9	420.8	166.4
	1600	31.7	28.0	7.4	787.4	713.7	202.8
CNF	200	23.8	22.6	11.4	88.4	76.4	39.6
	400	14.8	12.8	1.2	134.0	119.2	21.4
	800	7.3	6.5	0.0	180.0	151.2	8.6
	1600	3.4	2.6	0.0	212.0	173.2	4.6
MCNF	200	46.6	46.4	35.1	136.2	135.0	53.3
	400	41.8	41.7	27.8	247.7	252.1	130.4
	800	34.6	33.5	17.6	470.0	454.4	182.9
	1600	29.5	29.5	4.9	798.1	797.6	178.4
PAR4	200	46.6	40.6	38.6	151.6	104.6	98.0
	400	48.3	36.5	28.9	324.8	203.4	137.2
	800	44.9	7.3	5.3	647.4	140.6	70.0
	1600	41.6	3.1	3.8	1256.2	173.2	129.2
PAR5	200	51.4	51.5	48.2	162.0	124.6	114.8
	400	49.1	47.3	41.4	336.0	246.4	201.4
	800	46.6	40.0	34.8	649.3	428.0	316.0
	1600	45.8	7.5	15.6	1378.0	245.0	372.0

5 Summary

In this paper, we present a novel approach to feature construction. In this approach, a higher level attribute is constructed from two lower level attributes under the guidance of the distributions of positive and negative examples on the two lower level attributes. This approach is used to perform the task of feature construction in a general framework for concept learning with constructive induction. A simple prototype of the approach has been implemented and is presented in this paper. This prototype has been integrated into a decision tree learning system to perform constructive induction on decision trees. The experimental results empirically show that our approach outperforms the standard decision tree learning algorithm on all three domains tested.

The researches reported in [4,6,12] are most closely related to the work reported in this paper. In comparison with these related researches, the novel aspects of our work is summarized below. The approach presented in this paper is a general scheme to feature construction which can be integrated with rule learning systems, instance-based learning systems as well as decision tree learning systems, while the above mentioned related work was proposed for performing constructive induction on decision trees. Our approach is able to construct attributes with multi-values as well as binary attributes, while the related work only constructs binary attributes. In the related work, primitive attributes must be binary attributes, this assumption prevents the related work from applying to many real world applications. In our approach, primitive attributes may have multiple or even continuous values. Moreover, a value of a constructed attribute may represent a region which is not orthogonal to both axes; generalization may be performed on feature construction. Finally, the related work mentioned above can only detect conjunctive or disjunctive patterns, while our approach can detect many more different patterns. Disjunction, conjunction and exclusive-or are among these patterns.

One possible limitation is that the approach may not be able to detect any useful patterns when attributes are highly interactive, because the distributions of positive and negative examples may not reveal the correlation of two attributes. As indicated in [12], when concepts are hard, low-level primitive attributes tend to participate repeatedly in many high-level attributes. Therefore, this limitation may prevents the approach from constructing many useful high-level attributes when concepts are hard so as to limit the practical usefulness of the approach. One way to alleviate the problem is to construct different attributes under different conditions. For example, in a decision tree learning system, different attributes may be generated for different subtrees. Another possible way to overcome the problem is to use pattern guided approach discussed in section 2.3. Another limitation of the approach is, as shown in the experiments, that a large number of examples are needed in order to detect relevant patterns.

The work reported in this paper is only in its preliminary stage. Therefore, a lot of problems need to be addressed in the future. First, the prototype system needs to be extensively evaluated both theoretically and empirically. Empirically, the system should be tested on more problems including both artificial and practical problems, and needed to be compared with other related work, especially those reported in [4,6,12]. Theoretically, we need to show how, when and why this approach works or not.

Second, an advanced system needs to be developed. The advanced system must be able to construct attributes from linear and continuous attributes. The more advanced partition algorithm introduced in section 2.3 needs to be designed and incorporated in the advanced system. The advanced system must be capable of performing generalizations. That is, the partition is only applied to the points with significant amount of observed examples, and then a concept learning system is applied to generalize the subsets generated above.

Third, the approach needs to be integrated into a rule learning system, an instance-based learning system as well as a decision tree learning system. For each learning system, we need to invent a set of different heuristics for recommending pairs of attributes which are used to construct new attributes. Finally, the pattern guided and knowledge guided methods must be carefully investigated.

References

1. Clark, P., and Niblett, T. (1989). *The CN2 induction algorithm.* Machine Learning, 3.
2. Drastal, G., Czako, G., and Raatz, S. (1989). *Induction in an abstraction space: a form of constructive induction.* Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Deroit, MI.
3. Fayyad, U.M., Irani, K.B. (1992). *On the handling of continuous-valued attributes in decision tree generation.* Machine Learning 8(1).
4. Matheus, C. J. (1989). *Feature Construction: An Analytical Framework and an Application to Decision Trees.* PhD thesis, University of Illinois at Urbana-Champaign.
5. Michalski, R.S. (1983). *A theory and methodology of inductive learning.* In Machine learning: an artificial intelligence approach. Edited by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. Tioga Publishing Co., Palo Alto, CA.
6. Pagallo, G. (1990). *Adaptive Decision Tree Algorithms for Learning from Examples.* Ph.D thesis, University of California at Santa Cruz.
7. Quinlan, J.R. (1986). *Induction of decision trees.* Machine Learning 1(1).
8. Rendell, L.A., and Seshu, R. (1990). *Learning hard concepts through constructive induction: framework and rationale.* Computational Intelligence, 6.
9. Schlimmer, J.C. (1987). *Learning and representation change.* In proceedings of the Sixth National Conference on Artificial Intelligence.
10. Utgoff, P.E. (1986). *Shift of bias for inductive concept learning.* In Machine learning: an artificial intelligence approach, Vol. II. Edited by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. Tioga Publishing Co., Palo Alto, CA.
11. Utgoff, P.E. (1990). *An incremental method for finding multivariate splits for decision trees.* Proceedings of the Seventh International Conference on Machine Learning, Austin, TX.
14. Yang, D.S., Rendell, L.A., and Blix, G. (1991). *A Scheme for Feature Construction and a Comparison of Empirical Methods.* Proceedings of the Twelfth Internation Joint Conference on Artificial Intelligence, Sydney, Australia.
15. Wnek, J., and Michalski, R.S. (1993). *Hypothesis-driven constructive induction in AQ17: A method and experiments.* Machine Learning, Special Issue on Evaluating and Changing Representations.

Reasoning about Action and Time with Epistemic Conditionals

Nicholas Asher

IRIT - Université Paul Sabatier - 118 Route de Narbonne,
F-31062 Toulouse Cedex (France)
email: asher@irit.fr

1 Introduction

This paper investigates how *commonsense entailment* (CE) a conditional nonmonotonic logic introduced in Asher and Morreau (1991), can be used to formalize reasoning about action and change. To analyze problems of reasoning about change, CE is embedded within a multimodal framework encompassing dynamic operators for actions.

This approach employs a different strategy from that exploited in alternative ap proaches using circumscription, where nonmonotonic reasoning principles tailored to reasoning about change are developed externally to the logic–for instance, an externally imposed preference ordering on the models. From recent studies like those of Lifschitz and Rabinov (1988) and Sandewall (1993), it is becoming increasingly clear that there no single preference ordering that can be used to model all the aspects of commonsense reasoning about action. Once we admit several preferential relations, however, we have less than a clear idea of how to put them together. Further, such an approach in fact makes difficult the very thing we want to study–the logic of persistence and change; for it places the principles of persistence and causation, as well as their interaction, *outside* the logic. The interaction can only be studied indirectly and informally in the metalanguage, by means of ordering strategies, and these strategies are tedious to formalize and to study in an exact manner.

On my approach which uses an extension of the language of CE in which changes and actions can be represented, I provide axioms for persistence and a scheme for expressing commonsense causal laws. Within this framework, CE, a nonmontonic formalism developed for taxonomic reasoning, also suffices for treating a number of problems of reasoning about change– qualification, ramification, and surprises–and without externally imposed pref erential relations and without adding to the complexity of the basic nonmonotonic formalism of CE. This approach extends significantly the range of problems about action covered by circumscriptive approaches (for instance, Lin & Shoham, 1991), as theorems 7, 10 and 12 reveal.

2 The Language and Logic of Commonsense Entailment

The set of formulas of the language of CE, $\mathcal{L}_{\mathrm{CE}}$, is defined as the smallest set closed under the following rules:

- Propositional variables, $p, q, r, \ldots$ are formulas
- If ϕ and ψ are formulas, then so are $\neg\phi, \phi \wedge \psi, \phi > \psi$.

The other connectives, $\rightarrow, \wedge, \leftrightarrow$ are defined as usual.

The semantics of $\mathcal{L}_{\mathrm{CE}}$ is standard for conditional logic. Let $M = \langle S, *, [\,] \rangle$ be an $\mathcal{L}_{\mathrm{CE}}$ model, where S is a nonempty set of states,$[\,]$ is an interpretation function that assigns propositional variables truth values at states, and * is a selection function taking states and propositions (sets of states) into propositions–$* : S \times \wp(S) \rightarrow \wp(S)$ where $*$ has the following constraints:

- $*(s,p) \subseteq p$
- $*(s, p \cup q) \subseteq *(s,p) \cup *(s,q)$.

The semantics for conditional formulas $\phi > \psi$ is as one would expect: $M, s \models \phi > \psi$ iff $*(s, ||\phi||) \subseteq ||\psi||$, where $||\phi|| = \{s' : M, s' \models \phi\}$. The semantics for $\mathcal{L}_{\mathrm{CE}}$ is axiomatizable (Asher & Morreau, 1991).

It is convenient to add a modal operator $\Box$ to $\mathcal{L}_{\mathrm{CE}}$ in order to represent semantic entailments and "hard" taxonomic facts that interact with $>$ statements in nonmonotonic reasoning (Asher & Koons 1993, Asher in press b). We expand $\mathcal{L}_{\mathrm{CE}}$ models by adding an accessibility relation $R_{\Box}$; $R_{\Box}$ is an equivalence relation (S5) on the set of states.

The nonmonotonic consequence relation of commonsense entailment, $\approx\!\!|$, is defined using the canonical model CAN, whose set of worlds or states, S_{CAN}, consists of the maximal consistent sets of formulas. The basic idea (see Asher & Morreau 1991, Asher 1993, Asher in press b for further details) is that nonmonotonic inference from a set of premises Σ, which includes conditionals of the form $\phi > \psi$ that say what are the tendencies of normal ϕs (to ψ), involves two notions: (i) Σ encompasses all that one knows that is relevant to the deduction; (ii) one assumes that each item mentioned in Σ is as normal as the information in Σ allows it to be–what is normal for each item being dictated by the conditionals in Σ itself. To capture (i), $\approx\!\!|$ first considers all and only those worlds in S_{CAN} that verify Σ; this defines a set of worlds or an *information state* σ in which nothing more that Σ and its logical consequences are known. The second element of the characterization above is given by defining an operation of *normalization* with a parameter for a proposition. This operation applies to an information state such as σ; it eliminates from the information state all those worlds that are p worlds and not *normal* p worlds. The iterated application of normalization to σ, for each "relevant" proposition ϕ, gives rise to a *normalization sequence.* Our strategy is to define $\approx\!\!|$ in terms of the fixed points of the revision process. To ensure that the normalization sequences defined from an arbitrary set of premises reach a fixed point, we must define normalization sequences by transfinite recursion on the ordinals. The central definitions of normalization (NORM), normalization sequence and $\approx\!\!|$ are given below. Let $ANT(\Gamma) = \{\psi : \psi > \phi$ occurs a in subformula in $\Gamma\}$ and let $*(\sigma, p) = \bigcup_{s \in \sigma} *(s,p)$, and let $\mathrm{ANT}(\Gamma) = \{\psi : \psi > \varphi$ is a subformula of a formula in $\Gamma\}$.

Definition 1. Let $\sigma, p \subseteq \mathrm{S_{CAN}}$, the set of states in the canonical model.
$\mathrm{NORM}(\sigma, p) = \{s \in \sigma : s \notin p \backslash *(\sigma, p)\}$, if $\sigma \cap *(\sigma,p) \neq \emptyset$
$= \sigma$ otherwise.

Definition 2. $\Gamma(\zeta)$ the Γ-Normalization sequence for a given ordering ζ on $\mathrm{ANT}(\Gamma)$
$\Gamma_0(\zeta) \quad = \mathrm{W_{CAN}} \cap ||\Gamma||$
$\Gamma_{\beta+1}(\zeta) = \mathrm{NORM}(\Gamma_\beta(\zeta), p)$, for $\zeta(p) = n \;\&\; \beta + 1 = \lambda + m(n)$.
$\Gamma_\lambda(\zeta) \quad = \bigcap_{\beta<\lambda} \Gamma_\beta(\zeta)$

Definition 3. $\mathcal{B}$ is a Γ-fixpoint for a Γ normalization sequence $\Gamma(\zeta)$ iff $\exists\beta$ such that $\Gamma_\beta(\zeta) = \mathcal{B} \wedge \forall\alpha \geq \beta \Gamma_\beta(\zeta) = \Gamma_\alpha(\zeta)$.

Given that a normalization sequence is defined on all the ordinals, set theory shows that any normalization sequence beginning with an information state that is a set must eventually reach a fixed point in the sense defined.

Definition 4. $\Gamma \approx\!\!| \varphi$ iff for every Γ fixpoint $\mathcal{B}$, $\mathcal{B} \models \varphi$.

For a finite theory Γ, a Γ-fixpoint is reached in a finite number of normalizations and there are only finitely many of these fixed points. We can encode the fact that a Γ fixpoint

$\mathcal{B}_k$ verifies ϕ by means of a complex conditional $\Theta_k(\&(\Gamma), \phi)$; that ϕ is a nonmonotonic consequence of Γ, $\Gamma \approx\!\!| \phi$, is expressed by the conjunction $\Theta(\&(\Gamma), \phi)$ of the conditionals $\Theta_1(\&(\Gamma), \phi)$, $\Theta_2(\&(\Gamma), \phi)$,..., $\Theta_n(\&(\Gamma), \phi)$.[1]

3 Representing Change with Commonsense Entailment

To represent action and change, change, I use a fragment of dynamic logic.[2] I expand $\mathcal{L}_{\mathrm{CE}}$ to $\mathcal{L}_{\mathrm{CEA}}$ (CE-Action), which includes a finite set of atomic actions $\alpha_1, ..., \alpha_m$, the consecution operation, and the test operation ? on state formulas–where, if ϕ is a state formula then $?\phi$ is an action formula. I also introduce two operators $\mathcal{D}$ and $\mathcal{R}$ which give us the domain and the range of actions (the action is being done and the action has just been completed). They take action formulas into state formulas. The semantics of $\mathcal{L}_{\mathrm{CEA}}$is a modal extension of $\mathcal{L}_{\mathrm{CE}}$. A CEA model extends a CE model with a set of alternativeness relations R_α—one for each basic action. The semantic clauses needed for the interpretation of programs and new state formulas are:

$M, s, \models \mathcal{R}(\pi)$ iff $s \in Range(\mathcal{R}_\pi)$ $\quad$ $M, s \models \mathcal{D}(\pi)$ iff $s \in Domain(\mathcal{R}_\pi)$

$\mathcal{R}_{\pi_1;\pi_2} = \mathcal{R}_{\pi_1} \circ \mathcal{R}_{\pi_2}$ $\quad$ $\mathcal{R}_{?\phi} = \{\langle s, s\rangle : s \models \phi\}$

We can define rudimentary tense operators, Next (N) and Last (L), with these actions and operators, and, where δ is restricted to sequences of basic actions, restricted future and past tense operators.

$N\phi$ iff $\bigvee_{\alpha \in Basic-Action} \mathcal{D}(\alpha; ?\phi)$ $\quad$ $L\phi$ iff $\bigvee_{\alpha \in Basic-Action} \mathcal{R}(?\phi; \alpha)$

$F_\delta\phi$ iff $\mathcal{D}(\delta; ?\phi)$ $\quad$ $P_\delta\phi$ iff $\mathcal{R}(?\phi; \delta)$

Basic actions thus move time along one time unit. Since our models are classical, at each state we have: $N\neg\phi \leftrightarrow \neg N\phi$ and $L\neg\phi \leftrightarrow \neg L\phi$. Further, we have $\mathcal{D}(\alpha) \rightarrow F_\alpha \mathcal{R}F(\alpha)$ and $\mathcal{R}(\alpha) \rightarrow P_\alpha \mathcal{D}(\alpha)$.

From our semantics it follows that all temporally accessible states are alethically possible. So in addition to the basic axioms for modal, dynamic CE (CEA), we have:

(A1) $\Box\phi \rightarrow (F_\alpha\phi \wedge P_\alpha\phi)$

Again an axiomatization and a canonical model completeness proof is possible for this modal, action language, $\mathcal{L}_{\mathrm{CEA}}$, and its semantics.[3] In addition, I assume axioms ensuring the temporal homogeneity of commonsense laws:

(A2a) $F_\alpha(\phi > \psi) \rightarrow (F_\alpha(\phi) > F_\alpha(\psi))$

(A2b) $P_\alpha(\phi > \psi) \rightarrow (P_\alpha(\phi) > P_\alpha(\psi))$

By representing change and time, we can also represent causality and persistence. Causal laws in commonsense usually have exceptions though they hold across all epistemic possibilities, I will assume. So we should represent the causal relation between actions and their outcomes as a relation that holds *normally* or in *unexceptional circumstances.* Actions of one type tend normally to produce circumstances of the outcome type. Causal statements about action types are hence defeasible (Lewis 1973); doing α typically brings about ψ. Together with the dynamic logic modalities for actions, I use $>$ to express causal relations between and action α, preconditions ϕ and result ψ as follows: $\Box((\mathcal{D}(\alpha) \wedge \phi) > F_\alpha\psi)$.

[1] A proof-theoretic representation in Asher (1993b) permits the reduction for finite sets of premises of $\approx\!\!|$ to $\vdash_{CE}$ together with consistency checks and some other requirements.

[2] In Asher (1994), I used a version of the situation calculus to express somewhat similar ideas, but the extension with dynamic logic is more elegant.

[3] A much richer logic is shown complete in Asher & Koons (1993) and Koons & Asher (1994).

The persistence of a state through change is a disposition of the state, and like other dispositions is also expressed using $>$ (Asher & Morreau 1991). *Weak Persistence* says that a state ϕ typically persists, persistence being expressed by a conditional itself—$\phi > N\phi$.

- **Weak Persistence:** $\phi > (\phi > N\phi)$, where ϕ is a state formula.

From the A2 axioms, the axioms for N and L and the definition of weak persistence follows a backward persistence axiom: $L\phi > (L\phi > \phi))$

We can also express the interaction between the persistence of states and the causal effects of actions. A certain action may cause a change of state–from ϕ to $\neg\phi$–thus blocking the persistence of ϕ. But as we are modelling agents who reason about physical processes and their effects in situations where they only have partial information, we must allow that actions, though not understood as causing the termination of a given state, may be seen to call its persistence into question. For example, if a ball is rolling in linear fashion across a table and somebody suddenly knocks the table, we can with this description no longer predict what will happen to the ball. But we no longer expect that the ball will continue along its inertial path. When actions call into question the persistence of a state, I will say that they *affect* the state:[4]

Definition 5. $Affect(\alpha, \phi)$ iff $\Box(\mathcal{D}(\alpha) \rightarrow \neg(\phi > N\phi)) \wedge \Box(\mathcal{R}\alpha \rightarrow \neg(L\phi > \phi))$

If $Affect(\alpha, \phi)$, then $\mathcal{D}(\alpha)$ blocks the persistence of ϕ. This sort of failure of persistence is crucial to solving instances of the Frame Problem involving ramification, as we shall see. But problems of reasoning about action do not mention explicitly whether one event affects a state or not. So we must derive appropriate instances of $Affect(\alpha, \phi)$ from a typical description of a problem about reasoning about action, or, more precisely, how an agent might conceive of the problem given a certain description of it. To do this we must investigate the connection between Affect and the causal laws and integrity constraints the agent knows about. And to do this, I introduce below a belief operator and a relation of doxastic evidence.

4 Adding Belief and Evidence

The modal operator K (added to $\mathcal{L}_{\text{CEA}}$to form $\mathcal{L}_{\text{CEAK}}$) encodes at each state s the reflective beliefs of the agent at s and furnishes doxastic closure principles on a theory as does Moore's belief operator. The semantics for the belief operator is given by an alternativeness relation on states, which is normal, transitive and euclidean (K45). Typically the information then at the disposal of our reasoning agent will be of the form $\xi \wedge K(\xi)$–that is, he will have information that ξ and have reflected on ξ; we shall be interested what defeasible conclusions the agent draws from this information.

With the help of K, we can define an evidential operator $\mathcal{E}$ on pairs of propositions that will capture interactions between causal relations and persistence principles within an agent's set of beliefs. On the intended interpretation of the evidential relation $\mathcal{E}$ (ϕ, ψ), ϕ gives plausible grounds for ψ and on which ϕ for the agent constitutes the total evidence for ψ. I suppose that the agent's knowledge state is finitely axiomatizable and that it can be represented by a sentence in DNF of the form $\theta_1 \wedge ... \wedge \theta_n$.[5]

[4] Thanks to Andy Herzig and Luis Fariñas for conversations about interference that led to this notion of Affect.

[5] Since each conjunct is a dijunction of litterals or $>$ formulas, we can eliminate any quantification over propositions in the definition of $\mathcal{E}$.

Definition 6. $\mathcal{E}\ (\phi, \psi) \leftrightarrow (K(\phi > \psi) \wedge \bigwedge_{1 \leq i \leq n} \{K(\theta_i \leftrightarrow (\chi_i > \psi)) \rightarrow K(\chi \rightarrow \phi)\})$.

Now if it is epistemically possible to get the conditions under which a causal law of the form $(\mathcal{D}(\alpha) \wedge \phi) > F_\alpha\psi$ applies, then we should conclude that α affects $\neg\psi$. Assuming that our agent believes a set of facts and causal laws axiomatized by a sentence ξ and only these facts and laws, then the fact that $\Theta_j(\xi,\ phi)$ (that there is a ξ fixpoint that verifies ϕ) gives us the relevant sense of possibility of ϕ. Formalizing this axiomatically is (A3):

(A3) $K((\mathcal{D}(\alpha) \wedge \phi) > F_\alpha\psi) \wedge (\Theta_1(\xi, \phi) \vee \ldots \vee \Theta_n(\xi, \phi))) \rightarrow (K\xi \rightarrow Affect(\alpha, \neg\psi))$

Evidence and entailment also determine Affect. Whenever an action α affects the persistence of a state ϕ and ϕ is entailed by ψ, then α should affect also the persistence of ψ. For example, suppose there are two balls 1 and 2 at rest on a table at positions 1 and 2. If the action of moving the table typically moves ball 1 out of position 1, then we will not want to infer the persistence of ball 1 in position 1 and ball 2 in position 2. Thus we have the axiom, (A4):

(A4) $\{\Box(\chi \rightarrow \phi) \wedge Affect(\beta, \phi)\} \rightarrow Affect(\beta, \chi)$

(A4) does not capture the intuitive relation between Evidence and Affect by itself. If ϕ constitutes the evidence for χ, then if a particular action undermines the persistence of ϕ, then it should in general also undermine the persistence of χ. But the presence of causal laws can influence this relation. If one has a causal law about events of type α that bring about $\neg\chi$ but one cannot conclude that one of these events has occurred, then simply having the evidential connection that $\mathcal{E}\ (\phi, \chi)$ and the fact that $Affect(\beta, \phi)$ should not affect the persistence of χ. For instance, if one believes that turning on the switch turns the light on and the switch was last observed to be turned off, one will have evidence that the switch has been turned on if one now observes that the light is on. But if one has reasons to suppose that this action in fact did not occur (e.g. no person was observed near the switch), one will conclude that the light went on for some other reason and one will continue to infer the persistence of the position of the switch. On the other hand, if one knows that one of the events that brings about $\neg\chi$ has occurred, then in virtue of the evidential connection χ should be also affected. This is summed up in the two axioms of (A5). Again assume that the basic facts of agent's cognitive state are axiomatized by ξ, and that $K\xi$ means that the agent has reflected on this information. To represent the negative part of the claim we once again rely on the assumption that the agent's knowledge state is finitely axiomatizable and representable by a sentence in DNF $\theta_1 \wedge \ldots \wedge \theta_n$.

(A5.a) $\mathcal{E}\ (\phi, \chi) \rightarrow [K(((\mathcal{D}(\alpha) \wedge \psi) > F_\alpha\neg\chi) \wedge (\mathcal{D}(\beta) \rightarrow \mathcal{D}(\alpha))) \rightarrow [K\xi \rightarrow (Affect(\beta, \phi) \rightarrow Affect(\beta, \chi))]]$

(A5.b) $\mathcal{E}\ (\phi, \chi) \rightarrow \{\bigwedge_{1 \leq i \leq n} \neg K(\theta_i \rightarrow (\mathcal{D}(\alpha) \wedge \psi) > F_\alpha\neg\chi))) \rightarrow (K\xi \rightarrow (Affect(\beta, \phi) \rightarrow Affect(\beta, \chi)))\}$

(A5a-b) are useful in reasoning about problems of change with ramification. Consider, for example, a shower with a cold water pipe and hot water pipe joining together in a common pipe leading to the showerhead (the example is due to Sandewall). Consider now the following scenario describing an initial state or a shower and one causal law: $\neg shower$ (no water coming out of shower head); $shower \leftrightarrow (cold \vee hot)$ (water comes out of cold water pipe or out of hot water pipe); $\mathcal{D}(fix)$ (the plumber is fixing the shower); $\mathcal{D}(fix) \rightarrow N(shower)$ (fixing the shower makes it work)

Now you want it to be possible that you can take a shower that is neither ice-cold nor boiling hot. Persistence alone and exploitation of the causal law yields the wrong result: they predict that after the plumber has gone, you can only either take an ice-cold shower or a scalding hot one. The plumber evidently has not done his job! However, this problem has an important evidential connection:$\mathcal{E}$ ($\neg shower, \neg hot \wedge \neg cold$). Since there is no causal law about hot and cold that involves the plumber, then fixing the shower head will lead, using (A5.b) and the evidential connection just mentioned to $Affect(fix, \neg hot \wedge \neg cold)$ and then by the laws for $Affect$, we get $Affect(fix, \neg hot)$ and $Affect(fix, \neg cold)$. This results in the failure of either $\neg hot$ or $\neg cold$ to persist. So by adding axioms about evidence and about causal relations as with A3 and A5, you can get in this system more intended models; usually only the opposite is the case. The reflection on what one knows, represented by $K\xi$, may result in the agent's concluding less by default; for if the agent concludes an Affect proposition, he then refrains from inferring persistence by default.

5 APPLICATIONS OF THE THEORY

5.1 The Language A

CEAK (CE together with axioms A1-A5 and weak persistence) is provably correct for the class of actions whose intended outcomes are captured by the semantics of the language A of Gelfond and Lifschitz (1992). **A** is a simple language in which many well-known problems about change can be expressed and whose consequence notion yields the intended consequences of those problems. Given a set of litterals and a set of action terms A, the formulas of bf A to be either of the form $initially(\phi)$, ϕ after $A_1, ..., A_n$, or A_1 causes ϕ, if $\psi_1, ..., \psi_n$, where $A_1, ..., A_n \in A$ and $\phi, \psi_1, ..., \psi_m$ are litterals. It will be useful for later to look at a slight extension of **A** in which the $\phi, \psi_1, ..., \psi_m$ are allowed to be conjunctions of litterals.

Theorem 7. *Suppose that τ is a translation from A into $\mathcal{L}_{\mathrm{CEAK}}$as follows:*

- $\tau(initially(\neg)C) = (\neg)C$ *(C is an atomic formula).*
- $\tau(A_i) = \alpha_i$*, where α_i is a basic action symbol.*
- $\tau(\phi after A_1, ..., A_n) = F_{\alpha_1;\alpha_2;...;\alpha_n}\tau(\phi)$.
- $\tau(A$ *causes* ϕ *if* $\psi_1, ..., \psi_n) = \Box((\mathcal{D}(\alpha) \wedge \tau(\psi_1) \wedge ... \wedge \tau(\psi_n)) \rightarrow F_\alpha \tau(\phi))$.

Let T be a set of **A** *statements and let $\models_A$ be the A consequence relation, and let $\tau(T)*$ be the translation of T closed under the axioms CEAK. Then for every formula ϕ of the* **A** *language:*

$T \models_A \phi$ *iff* $K\tau(T*) \wedge \tau(T*) \mid\!\approx \tau(\phi)$.

This fact, whose proof had to be omitted because of space limitations (and which proceeds by an examination of cases and in the case of formulas of the form $\phi after A_1, ..., A_n$ exploits an induction over transitions between states), establishes that CEAK is sound and complete with respect to $\models_A$ and is at least as adequate as those theories discussed in Kartha (1993)

5.2 Indeterminate Actions with Realistic Causal Laws

CEAK handles problems beyond the class describable with A. For example, while **A** only handles laws about actions that admit of no exceptions, CEAK also handles more realistic causal laws. In the standard formalization of the Yale Shooting Problem, the relevant causal laws would be expressed in CE as strict conditionals. which reflects an implausible understanding of causal laws. More plausible is a reformulation of causal laws using $>$. Here is the Yale Shooting Problem with a realistic causal law (realistic YSP): $\Box((\mathcal{D}(shoot) \wedge loaded) > N\neg alive)$; $alive \wedge loaded\ \mathcal{D}(wait)$; $N(\mathcal{D}(shoot))$.

Let $\mathcal{L}_{\mathrm{CEAK}}$restricted to the atoms $\mathcal{D}(shoot)$, *loaded*, *alive*, $\mathcal{D}(wait)$ and Boolean combinations of these formulas up to within the scope of two iterations of N be denoted $\mathcal{L}_{\mathrm{CEAK}}|_{YSP}$.

Theorem 8. *Let ξ be the conjunction of the facts for realistic YSP together with the axioms and instances of the axioms of CEAK restricted to $\mathcal{L}_{\mathrm{CEAK}}|_{YSP}$. Then $\xi \wedge K(\xi) \mid\!\approx NN\neg alive$.*

The proof of this proposition (again omitted because of space limitations) exploits the link between the causal laws and the Affect operator. First to form our basic state, we take all those worlds in the canonical model which ξ is believed and in which ξ. This means taking all the worlds consistent with ξ and the same set as the set of belief alternatives (which implies that the agent only knows ξ). By normalizing first on the proposition *loaded* twice, we get $Nloaded$. Thus, there is a provable conditional, $\Theta_j(\xi, Nloaded)$, whose terms are all consistent with ξ. Now a particular instance of the causal law $(\mathcal{D}(shoot) \wedge loaded) > N\neg alive$ can be used in conjunction with rule (A3) to conclude: $Affect(shoot, alive)$. This now allows us to infer monotonically together with the axioms for the interaction of N and $>:\neg(N(alive) > NN(alive))$. This blocks any application of the persistence axioms to yield NNalive under normalization. We then can verify that in every ξ fixpoint that $NN\neg alive$ (equivalently, $negNNalive$) by means of a so called "survivor proof" in commonsense entailment. The idea is that we show that there is a world which verifies every sentence in ξ, verifies $NN\neg alive$ and survives all normalizations in any order. We then show that the presence of "this survivor world" in every ξ fixpoint ensures that every fixpoint verifies $NN\neg alive$.

The following proposition, whose proof can be reconstructed given the techniques for proving the defeat of defeasible modus ponens for > detailed in Morreau (1992), shows that this formalism is more resourceful than other frameworks–including **A**.

Proposition 9. *: Suppose that to the ξ of theorem 9 we add the proposition $NNalive \wedge Nloaded$ and call the result $\xi*$. Then: $not(K(\xi*) \wedge \xi* \mid\!\approx NN\neg alive)$.*

The premises of this proposition represent a plausible scenario when speaking of real actions. Suppose the gun is loaded and is fired but the intended target does not die. We do not co nclude in this case inconsistency but rather that we have some abnormal situation with respect to the causal law–we missed the target, the bullet failed to cause sufficient injury, or etc. Thus, in the classification of Sandewall (1993), $\mathcal{L}_{\mathrm{CEAK}}$thus handles at least some qualification problems in the IAN class. Here is a more elaborate example of a qualification problem that comes from the literature on causation. If you strike the match, it will light. If you strike the match and it is wet, it won't light. I am told that X will strike the match what can I predict? Here we have two defeasible causal rules– one of which has a more specific antecedent than the other, when they are naturally formalized in $\mathcal{L}_{\mathrm{CEAK}}$.

- $Match(x) \wedge \mathcal{D}(strike(x)) > \mathcal{F}_{strike(x)} light(x)$
- $Match(x) \wedge wet(x) \wedge \mathcal{D}(strike(x)) > \mathcal{F}_{strike(x)} \neg light(x)$

Intuitively I should conclude that the match will light. But if I know in addition that it is wet, I conclude that it will not light. Any nonmonotonic formalism in which a more specific rule overrules a less specific one when the more specific antecedent is verified–and CE is such a formalism–will arrive at these conclusions. Generalizing from this example, suppose that $\models_{A*}$ is defined so that if we have an A-theory T with a set of causal laws $\{C_1, ..., C_n\} \subseteq T$ such that (i) their translations as defined below yield > conditionals with antecedents $\phi_1, ..., \phi_n$, for each i $\phi_{i+1} \rightarrow \phi_i$ and conclusions $\psi_1, ..., \psi_n$ such that $\Box(\psi_{i+1} \rightarrow \neg\psi_i)$ for each i and (ii) $T \models phi_j$ for $j > 1$, then the transition function of every admissible T structure satisfies C_j and not C_k for $k < j$. Then

Theorem 10. *Suppose that τ is a translation from* **A** *into $\mathcal{L}_{\mathrm{CEAK}}$as in theorem 8 except that $\tau(A\ causes\ \phi\ if\ \psi_1, ..., \psi_n) = ((\mathcal{D}(\alpha) \wedge \tau(\psi_1) \wedge ... \wedge \tau(\psi_n)) > F_\alpha \tau(\phi))$. Let T be a set of A statements and let $\models_A$ be the* **A** *consequence relation, and let $\tau(T)*$ be the translation of T closed under the axioms of CEAK. Then for every formula ϕ of the* **A** *language: $T \models_{A*} \phi \Rightarrow K\tau(T*) \wedge \tau(T*) \approx\!\!\!\!\!\models \tau(\phi)$.*

5.3 Indeterminate Actions with Ramifications

While the shower example of section 4 is one example of reasoning about action with ramifications (Sandewall's (1993) IAD class), another, more recalcitrant is due to Lifschitz (1990). Lifschitz imagines a lamp connected to two switches; the light goes on whenever the two switches are both up or on or both down or off. Suppose that you know that switch one is up, switch two is down and that someone moves switch 2. The action is nondeterministic, because it is at least possible that by the first action that switch 1 change its position. There are two possible outcomes, but one is highly preferred. CEAK predicts one outcome because of the causal laws the agent believes. Here is the $\mathcal{L}_{\mathrm{CEAK}}$formalization with the causal laws made explicit: $sw1 \wedge \neg sw2$; $\Box(\mathcal{D}(move(sw1(2))) > (\neg sw1(2) \rightarrow Nsw1(2)))$; $\Box(L \leftrightarrow (sw1 \leftrightarrow sw2))$; $\mathcal{D}(move(sw2)$. Let $\mathcal{L}_{\mathrm{CEAK}}|_{LL}$ be the restriction of $\mathcal{L}_{\mathrm{CEAK}}$to Boolean combinations of the atomic formulae above up to within the scope of one N operator. Now assume that the agent only knows the conjunction ξ of these facts.[6]

Theorem 11. *: Let ξ be the conjunction of the statements of the facts known to the agent in the Lifschitz lamp example together with the axioms and schemata of CEAK restricted to $\mathcal{L}_{\mathrm{CEAK}}|_{LL}$. Then $K(\xi) \wedge \xi \approx\!\!\!\!\!\models N(L \wedge sw1 \wedge sw2)$.*

To show this, note that from the integrity constraint $\Box(L \leftrightarrow (sw1 \leftrightarrow sw2))$ and the axioms for CE$(sw1 \wedge \neg sw2) > \neg L$ and $(sw2 \wedge \neg sw1) > \neg L$. Since there is no ϕ such that $\vdash \xi \rightarrow (\phi > \neg L \wedge \neg(\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge \neg sw1)))$,

$\{\neg(\phi > \neg L \wedge \neg(\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge \neg sw1))), \xi \wedge K(\xi)\}$ is consistent. So by the construction of the canonical model, we know that there is at least one world w, verifying $\xi \wedge K(\xi)$ such that $w \models \neg(\phi > \neg L \wedge \neg(\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge\ negsw1)))$. So we have in the basic information state: $\neg K[(\phi > \neg L \wedge \neg(\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge \neg sw1)))]$ for every ϕ and hence for every ϕ such that for some basic conjunct δ in the DNF equivalent of ξ, δ is logically equivalent to $\phi > \neg L$.

By the logic of the K operator (K45), $K\neg[(\phi > \neg L \wedge \neg(\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge \neg sw1)))]$, and so: $K[(\phi > \neg L) \rightarrow (\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge \neg sw1)))]$.

Now suppose $K(\phi > \neg L)$. Then, $K(\phi \rightarrow ((sw1 \wedge \neg sw2) \vee (sw2 \wedge \neg sw1)))$.

But since $K(sw1 \wedge \neg sw2)$, $K(\phi \rightarrow (sw1 \wedge \neg sw2))$. So $K(\phi > \neg L) \rightarrow K(\phi \rightarrow ((sw1 \wedge \neg sw2)))$, for every ϕ such that for some basic conjunct δ in the DNF equivalent of $\xi\delta$ is logically equivalent to $\phi > \neg$L.

By the definition of evidence, then: $\mathcal{E}\ (sw1 \wedge \neg sw2, \neg L)$.

Since $\mathcal{D}(move(sw2) > sw2$, then by (A3): $Affect(move(sw2), \neg sw2)$.

By (A4) and (A5): $Affect(move(sw2), sw_1 \wedge \neg sw2); Affect(move(sw2), \neg L)$.

By the definition of Affect: $\neg L > N \neg Land \neg((sw_1 \wedge \neg sw2) > N(sw_1 \wedge \neg sw2))$

By using the survivor world proof technique again (see theorem 8), we see that normalization on Σ now yields $N(sw1)$ from the persistence axiom $sw1 > (sw1 > Nsw1)$, while from the causal law we get: $N(sw2)$. The inference of $N\neg L$ and $N\neg sw2$ is blocked by the consequences of $Affect$. Note that even though by the integrity constraints and the reasoning above $\mathcal{E}\ (\neg L \wedge \neg sw2, sw1)$, we have $\neg Affect(move(sw2), sw1)$, because of the presence

[6] This is a counterexample to Winslett's method of updating (Asher, in press a) and also Baker's method of reasoning about action (Lifschitz 1990).

of the causal law about $sw1$ and the fact that we cannot from the facts known by the agent derive $move(sw1)$. By the axioms for N, $N(sw1 \wedge sw2)$. By the CE axioms and the integrity constraint, NL.

Corollary 1 *: Suppose we remove from ξ the statement of the causal laws and call the result ξ' . Then: $K(\xi') \wedge \xi' \not\mid\!\approx NL$ but $K(\xi') \wedge \xi' \mid\!\approx NL \vee N\ negsw1$.*

The corollary shows that the causal laws are crucial to a resolution of the problem.

To get a correctness proof for ramification problems, I extend the **A** language. I specify a set of affectable fluents for each action by introducing a predicate Affect taking an action symbol and a V-proposition of **A** as arguments and yielding a V-proposition of an extended language **A+**. The semantics for **A+** is just like that for **A**, except that Affect will constrain the persistence clause of propositions in an **A+** structure. An **A+** structure for a theory T is a pair of a set of states and a transition relation such that: (i) the transition relation satisfies (using the A definition) the action laws deducible from T; (ii) Every state in S satisfies the integrity constraints in T (using the usual satisfiability notion of propositional logic); (iii) V-propositions are true of the appropriate states (again using the **A** definition); (iv) if $\sigma \models \phi$ and $\sigma \not\models Affect(A, \phi)$ and (either $\sigma \not\models Causes(A, \neg\phi)$, if $P_1, ..., P_m$ or $\sigma \not\models P_1, ..., P_m$), then $\forall \sigma' \in A(\sigma) \sigma' \models \phi$; (v) $\sigma \models Affect(A, \phi)$ and ($\sigma \not\models Causes(A, \neg\phi)$, if $P_1, ..., P_m$ or $\sigma \not\models P_1, ..., P_m$)) and ($\sigma \not\models Causes(A, \phi)$, if $P_1, ..., P_m$ or $\sigma \not\models P_1, ..., P_m$)), then $A(\sigma) \cap \{\sigma' : \sigma' \models \phi\} \neq \emptyset$ and $A(\sigma) \cap \{\sigma' : \sigma' \models \neg\phi\} \neq \emptyset$. This last condition ensures that our models are maximal in a particular sense so that Affect cancels out persistence.

To show a correspondence between **A+** and $\mathcal{L}_{\text{CEAK}}$, I propose the same translation from A+ into $\mathcal{L}_{\text{CEAK}}$as from A except that $\tau(Affect(A, \psi)) = Affect(\tau(A), \tau(\psi))$. Then if $T*$ is the closure of $\tau(T)$ under the CEAK axioms,

Theorem 12. *Suppose T is a theory in the language of* A+. *$T \models_{A+} \phi$ iff $T * \wedge K(T*) \mid\!\approx \tau(\phi)$.*

The proof of theorem 12 follows the outlines of the proof of theorem 7.

In conclusion, I view reasoning about change as a special case of a general system of nonmonotonic reasoning, as is taxonomic reasoning; the general nonmonotonic consequence notion is adapte d to reasoning about change by choosing particular axioms about notions like causation and evidence which can be expressed in the rich language of dynamic logic. Interestingly, the construction of $\mid\!\approx$ and the decision problem for CEAK validity is no worse than that for CE alone. Given the proof theoretic representation of $\mid\!\approx$, the implementation of a $\mid\!\approx$ prover is feasible. But the main advantages of this approach are theoretical. There is an automatic interaction between reasoning about change and other forms of nonmonotonic reasoning. Other approaches like Sandewall's or Shoham's take reasoning about change to be a distinctive form of nonmonotonic reasoning, warranting in the end several special minimization policies in the process of picking preferred models, which complicates the integration of reasoning about change with other forms of nonmonotonic reasoning. Finally, the examples above show how to derive appropriate instances of the Affect relation from more general principles and descriptions. This presents an improvement on recent solutions to such problems about reasoning about action, in which one writes down in the coding of the problem the appropriate instances of Affect without deriving them from the statement of the problem (e.g., Lifschitz 1990).

References

1. Asher, N. (in press a): 'Belief Dynamics in a Changing World', Proceedings of ICCS 93, San Sebastian, Spain, Kluwer Academic Publishers.
2. Asher, N. (in press b): 'Commonsense Entailment: A Conditional Logic for Some Generics', in Crocco G., L. Farinas, A. Herzig, *Conditional Logics and Artificial Intelligence*, Oxford University Press.
3. Asher, N. (1994): Problems with Persistence', *Topoi*, 37-49.
4. Asher, N. (1993b): 'Extensions for Commonsense Entailment', Workshop on Conditionals, *Proceedings Of The 13th International Joint Conference On Artificial Intelligence*, Chambery, France.
5. Asher, N. & R. Koons (1993): 'The Revision of Beliefs and Intentions in a Changing World', *AAAI Spring Symposium Series On Reasoninag About Mental States: Formal Theories And Applications*, Stanford CA.
6. Asher, N. & M. Morreau (1991): 'Commonsense Entailment: A Modal Theory of Nonmonotonic Reasoning', *Proceedings Of The 12th International Joint Conference On Artificial Intelligence*, Sydney Australia.
7. Baker, A. (1991): 'Nonmonotonic Reasoning in the Framework of the Situation Calculus', *Artificial Intelligence* 49.
8. Gelfond, M.& V. Lifschitz (1992): 'Describing Action and Change by Logic Programs', *Proceedings Joint International Conference And Symposium On Logic Programming.*
9. Lewis, D. (1973): 'Causation', *Journal of Philosophy* 70, 556-567.
10. Lifschitz, V. (1987): 'Formal Theories of Action', *The Frame Problem In Artificial Intelligence*, pp. 35–57. Los Altos: Morgan Kaufman.
11. Lifschitz, V. & A. Rabinov (1990): 'Miracles in Formal Theories of Action', *Artificial Intelligence*,38, pp. 225–237.
12. Lifschitz, V. (1990): 'Frames in the Space of Situations', *Artificial Intelligence* 39.
13. Lin, F. & Y. Shoham (1991): 'Provably Correct Theories of Action: Preliminary Report', *AAAI* 91, Anaheim, CA.
14. Kartha,N. (1993): 'Formalization of Action', *Proceedings Of The 13th International Joint Conference On Artificial Intelligence*, Chambery, France.
15. Koons, R. & N. Asher (1994): 'Belief Revision in a Changing World',*TARK* 94.
16. Sandewall E. (1993): 'The Range of Applicability of Nonmonotonic Logics for the Inertia Problem', *Proceedings Of The 13th International Joint Conference On Artificial Intelligence*, Chambery, France.
17. Winslett, M. (1988): 'Reasoning about Action using a Possible Models Approach', in *Proceedings Of The Seventh National Conference On Artificial Intelligence*, 89–93.
18. Winslett, M. (1990): 'Sometimes Updates are Circumscription', *AAAI* 90.

The Semantics of Propositional Contexts

Saša Buvač, [1] Vanja Buvač, [2] and Ian A. Mason[3]

[1] Computer Science Department, Stanford University, Stanford, California 94305-2140, USA. buvac@sail.stanford.edu.
[2] HB 455, Dartmouth College, Hanover, New Hampshire 03755. vanja@dartmouth.edu.
[3] Computer Science Department, Stanford University, Stanford, California 94305-2140. iam@sail.stanford.edu.

Abstract. In this paper we investigate the semantic properties of *contexts*. We describe the syntax and semantics of the propositional logic of context. This logic extends classical propositional logic in two ways. Firstly, a new modality, $\mathtt{ist}(\kappa, \phi)$, is introduced. It is used to express that the sentence, ϕ, holds in the context κ. Secondly, each context has its own vocabulary, i.e. a set of propositional atoms which are *defined* or *meaningful* in that context. The main results of this paper are a proof that our logic is decidable and comparison of our semantics to Kripke semantics.

1 Introduction

In this paper we investigate the semantic properties of *contexts* as they appear in declarative AI. Contexts were first suggested in McCarthy's Turing Award Paper, [5], as a possible solution to the problem of generality in AI. Our main motivation for formalizing contexts is to solve this problem. We want to be able to make AI systems which are never permanently stuck with the concepts they use at a given time because they can always transcend the context they are in. Such a capability would allow the designer of a reasoning system to include only such phenomena as are required for the system's immediate purpose, retaining the assurance that if a broader system is required later, "lifting axioms" can be devised to restate the facts from the narrow context in the broader context with qualifications added as necessary. We provide two simple examples.

The first example is due to McCarthy [7]. It illustrates how a reasoning system can utilize contexts to incorporate information from a general common sense knowledge base into other specialized knowledge bases. Assume that in the context of situation calculus $on(x, y, s)$ is used to express the fact that object x is on top of object y in situation s. Although no mention to the notion of *above* is made in the context of situation calculus, we are interested to know which of the *above* relations hold in a particular situation. The definition of *above* in terms of *on* is likely to be found in some context of general common sense knowledge. The context formalism will allow a reasoning system to use the theory of situation calculus and the theory of general common sense knowledge together. Furthermore, in the logic we can write axioms to import or *lift* the definition of *above* from the context of general common sense knowledge into the context of situation calculus. Although *above* was not originally defined in the context of situation calculus, the system, after lifting, will be able to infer which *above* relations hold in a particular situation. Of course, the power of a full quantificational logic will be needed to adequately address this example.

The second example concerns theories which were not originally intended to be used together, and in fact might, on the surface, seem inconsistent. For example, assume a common sense knowledge base of Stanford University contains the proposition "kids drive BMW's". A common sense knowledge base of Berkeley, which was not originally intended to be used with the above mentioned Stanford knowledge base, will probably contain the negation of this proposition. A logic of context will enable a reasoning system to use such seemingly inconsistent knowledge bases without deriving a contradiction.

Although the notion of context has existed in philosophy, linguistics, and natural language processing for some time, our work is based on the idea of treating context as a *formal object.* This means that a context is an object in the semantics denoted by a corresponding constant in the language. The language also contains the `ist` modality (pronounced as "is true"), which allows us to talk about a sentence being true in a context. For example, $\mathtt{ist}(\kappa, p)$, means that the sentence p is true in the context κ. In the semantics, contexts are associated with a set of truth assignments, reflecting the states of affairs in that context. Proposition p is true in context κ if every truth assignment associated with κ satisfies p.

A formal logical explication of contexts was first given in [4]. In that paper we describe both the syntax and semantics of a general propositional language of context, and give a Hilbert style proof system for this language. The main results of that paper were the soundness and completeness of the proof system. We also provided soundness and completeness results (i.e. correspondence theory) for various extensions of the basic system.

The aim of this paper is to answer some semantic questions not resolved by the above mentioned paper. These are: (1) What is the relation between the semantics of context and Kripke semantics? (2) Is the system decidable, i.e. does the logic have the finite model property? Due to lack of space, proofs of the theorems are not included in this paper. These, as well as additional discussions on the subject and related works can be found in [3].

1.1 Notation

We use standard mathematical notation. If X and Y are sets, then $X \to_{\mathrm{p}} Y$ is the set of partial functions from X to Y. $\mathbf{P}(X)$ is the set of subsets of X. X^* is the set of all finite sequences, and we let $\bar{x} = [x_1, \ldots, x_n]$ range over X^*. ϵ is the empty sequence. We use the infix operator $*$ for appending sequences. We make no distinction between an element and the singleton sequence containing that element. Thus we write $\bar{x} * x_1$ instead of $\bar{x} * [x_1]$. As is usual in logic we treat X^* as a tree (that grows downward). $\bar{x}_1 < \bar{x}_0 \leq \epsilon$ iff $\bar{x}_1$ properly extends $\bar{x}_0$ (i.e.

$(\exists \bar{y} \in X^* - \{\epsilon\})(\bar{x}_1 = \bar{x}_0 * \bar{y}))$. We say $Y \subseteq X^*$ is a subtree rooted at $\bar{y}$ to mean (1) $\bar{y} \in Y$ and $(\forall \bar{z} \in Y)(\bar{z} \leq \bar{y})$ and (2) $(\forall \bar{z} \in Y)(\forall \bar{w} \in X^*)(\bar{z} \leq \bar{w} \leq \bar{y} \rightarrow \bar{w} \in Y)$.

2 The Propositional Logic of Context

The propositional logic of context extends classical propositional logic in two ways. Firstly, a new modality, $\mathtt{ist}(\kappa, \phi)$, is introduced. It is used to express that the sentence, ϕ, holds in the context κ. Secondly, each context has its own vocabulary, i.e. a set of propositional atoms which are *defined* or *meaningful* in that context. The vocabulary of one context may or may not overlap with another context.

2.1 Syntax

We begin with two distinct countably infinite sets, $\mathbb{K}$ the set of all contexts, and $\mathbb{P}$ the set of propositional atoms. The set, $\mathbb{W}$, of well-formed formulas (wffs) is built up from the propositional atoms, $\mathbb{P}$, using the usual propositional connectives (negation and implication) together with the ist modality.

Definition ($\mathbb{W}$): $\mathbb{W} = \mathbb{P} \cup (\neg\mathbb{W}) \cup (\mathbb{W} \rightarrow \mathbb{W}) \cup \mathtt{ist}(\mathbb{K}, \mathbb{W})$
The operations $\wedge$, $\vee$ and $\leftrightarrow$ are defined as abbreviations in the usual way.

We have chosen to develop a modal logic rather than to reifying sentences and treat ist as a regular predicate, because (1) it leads to a more natural semantics (defined in the following subsection), and (2) the language does not allow self-referential statements, thus avoiding paradoxes. Although self-referential formulas are relevant for developing theories of truth, they are not needed for describing states of affairs which hold in particular contexts. Therefore, the loss of expressive power due to lack of self-referential formulas will not be missed in our logic. The latter approach, of reifying formulas, is taken by Attardi and Simi in [1]. We further discuss their work in and its relation to our system in [3].

2.2 Semantics

To explain the semantics we first introduces a naïve notion of a model, which is then refined in two stages.

Naïvely, a context is modelled by a set of truth assignments, that describe the possible states of affairs of that context. Thus a model will associate a set of truth assignments with every context. These truth assignments reflect the states of affairs which are possible in a context. For a proposition to be true in a context it has to be satisfied by all the truth assignments associated with that context. Therefore, the ist modality is interpreted as validity: $\mathtt{ist}(\kappa, \rho)$ is true iff the propositional atom ρ is true in all the truth assignments associated with context κ. Treatment of ist as validity corresponds to Guha's proposal for context semantics, which was motivated by the Cyc knowledge base. A system which models a context by a single truth assignment, thus interpreting ist as truth, can be obtained by placing simple restrictions on the definition of a model and by enriching the set of axioms.

However, this naïve model is not powerful enough to represent some properties desired of contexts. Therefore, we need to refine our naïve notion of a model. We do this in two stages.

Firstly, the nature of a particular context may itself be context dependent. For example, in the context of the 1950's, the context of car racing is different from the context of car racing viewed from today's context. This naturally leads to considering sequences of contexts rather than a context in isolation. So, a model will associate a set of truth assignments with a context sequence, rather than an individual context (as was the case in the naïve view). We refer to this feature of the system as *non-flatness*. It reflects on the intuition that what holds in a context can depend on how this context has been reached, i.e. from which perspective it is being viewed. For example, non-flatness will be desirable if we represent the beliefs of an agent as the sentences which hold in a context. A system of flat contexts can easily be obtained by placing certain restrictions on what kinds of structures are allowed as models, as well as enriching the axiom system (cf. §3.2 in [3]).

Secondly, since different contexts can have different vocabularies, some propositions can be meaningless in some contexts, and therefore the truth assignments describing the state of affairs in that context need to be partial.

Now we are ready to define the general model:

Definition ($\mathfrak{M}$): In this system a model, $\mathfrak{M}$, will be a function which maps a context sequence $\overline{\kappa} \in \mathbb{K}^*$ to a set of partial truth assignments,

$$\mathfrak{M} \in (\mathbb{K}^* \rightarrow_{\mathrm{p}} \mathbf{P}(\mathbb{P} \rightarrow_{\mathrm{p}} 2)),$$

with the added conditions that

1. $(\forall \overline{\kappa})(\forall \nu_1, \nu_2 \in \mathfrak{M}(\overline{\kappa}))(\mathrm{Dom}(\nu_1) = \mathrm{Dom}(\nu_2))$
2. $\mathrm{Dom}(\mathfrak{M})$ is a subtree of $\mathbb{K}^*$ rooted at some context sequence $\overline{\kappa}_0$.

We write $\overline{\kappa}^{\mathfrak{M}}$ to denote the set of partial truth assignments $\mathfrak{M}(\overline{\kappa})$. Note that $\overline{\kappa}^{\mathfrak{M}}$ can be empty. Since all the elements of $\mathfrak{M}(\overline{\kappa})$ have the same domain, which is imposed by condition 1. above, we will write $\mathrm{Dom}(\mathfrak{M}(\overline{\kappa}))$ to refer to this domain. The collection of all such models will be denoted by $\mathbb{M}$.

Vocabularies To capture the intuition that different contexts can have different vocabularies, we make the truth assignments in our model partial. The atoms which are given a truth value in a context sequence are defined by a relation $\mathsf{Vocab} \subseteq \mathbb{K}^* \times \mathbb{P}$. Given a Vocab, the *vocabulary of a context sequence* $\overline{\kappa}$, or the set of atoms which are meaningful in that sequence, is $\{\rho \mid <\overline{\kappa}, \rho> \in \mathsf{Vocab}\}$.

Definition (*Vocab* of $\mathfrak{M}$): We define a function $\mathit{Vocab} : \mathbb{M} \rightarrow \mathbf{P}(\mathbb{K}^* \times \mathbb{P})$, which given a model returns the vocabulary of the model:

$$\mathit{Vocab}(\mathfrak{M}) := \{<\overline{\kappa}, \rho> \mid \overline{\kappa} \in \mathrm{Dom}(\mathfrak{M}) \text{ and } \rho \in \mathrm{Dom}(\mathfrak{M}(\overline{\kappa}))\}$$

We say that a model $\mathfrak{M}$ is *classical on vocabulary* Vocab iff Vocab $\subseteq$ *Vocab*$(\mathfrak{M})$.

The notion of vocabulary can also be applied to sentences. Intuitively, the vocabulary of a sentence relates a context sequence to the atoms which occur in the scope of that context sequence. In the definition we also need to take into account that sentences are not given in isolation but in a context.

Definition (Vocab of ϕ in $\overline{\kappa}$): We define a function Vocab : $\mathbb{K}^* \times \mathbb{W} \rightarrow \mathbf{P}(\mathbb{K}^* \times \mathbb{P})$ which given formula in a context, returns the vocabulary of the formula.

$$\mathrm{Vocab}(\overline{\kappa}, \phi) = \begin{cases} \{<\overline{\kappa}, \phi>\} & \phi \in \mathbb{P} \\ \mathrm{Vocab}(\overline{\kappa}, \phi_0) & \phi \text{ is } \neg\phi_0 \\ \mathrm{Vocab}(\overline{\kappa} * \kappa, \phi_0) & \phi \text{ is } \mathtt{ist}(\kappa, \phi_0) \\ \mathrm{Vocab}(\overline{\kappa}, \phi_0) \cup \mathrm{Vocab}(\overline{\kappa}, \phi_1) & \phi \text{ is } \phi_0 \rightarrow \phi_1 \end{cases}$$

It is extended to sets of formulas as follows:

$$\mathrm{Vocab}(\overline{\kappa}, \mathrm{T}) = \bigcup_{\phi \in \mathrm{T}} \mathrm{Vocab}(\overline{\kappa}, \phi).$$

Note that it is only in the propositional case that we can carry out this *static* analysis of the vocabulary of a sentence. This will not be possible in the quantified versions. Also note that our definition of vocabulary of a sentence is somewhat different from Guha's notion of definedness. Guha proposes to treat $\mathtt{ist}(\kappa, \phi)$ as false if ϕ is not in the vocabulary of the context κ.

Satisfaction We can think of partial truth assignments as total truth assignments in a three-valued logic. Our satisfaction relation then corresponds to Bochvar's three valued logic, [2], since an implication is meaningless if either the antecedent or the consequent are meaningless. We chose Bochvar's three valued logic because we intend meaningfulness to be interpreted as syntactic meaningfulness, rather then semantic meaningfulness which could be ascribed to Kleene's three valued logic.

Definition ($\models$): If $\nu \in \overline{\kappa}^{\mathfrak{M}}$ and $\mathrm{Vocab}(\overline{\kappa}, \chi) \subseteq$ *Vocab*$(\mathfrak{M})$, then we define satisfaction, $\mathfrak{M}, \nu \models_{\overline{\kappa}} \chi$, inductively on the structure of χ as follows:

$\mathfrak{M}, \nu \models_{\overline{\kappa}} \rho$ iff $\nu(\rho) = 1, \quad \rho \in \mathbb{P}$
$\mathfrak{M}, \nu \models_{\overline{\kappa}} \neg\phi$ iff not $\mathfrak{M}, \nu \models_{\overline{\kappa}} \phi$
$\mathfrak{M}, \nu \models_{\overline{\kappa}} \phi \rightarrow \psi$ iff $\mathfrak{M}, \nu \models_{\overline{\kappa}} \phi$ implies $\mathfrak{M}, \nu \models_{\overline{\kappa}} \psi$
$\mathfrak{M}, \nu \models_{\overline{\kappa}} \mathtt{ist}(\kappa_1, \phi)$ iff $\forall \nu_1 \in (\overline{\kappa} * \kappa_1)^{\mathfrak{M}} \quad \mathfrak{M}, \nu_1 \models_{\overline{\kappa} * \kappa_1} \phi$

If the preconditions $\nu \in \overline{\kappa}^{\mathfrak{M}}$ and $\mathrm{Vocab}(\overline{\kappa}, \chi) \subseteq$ *Vocab*$(\mathfrak{M})$ do not hold, then neither $\mathfrak{M}, \nu \models_{\overline{\kappa}} \chi$ nor $\mathfrak{M}, \nu \models_{\overline{\kappa}} \neg\chi$.

In the ist clause of the satisfaction relation note that $\overline{\kappa} * \kappa_1 \in \mathrm{Dom}(\mathfrak{M})$ since $\mathrm{Vocab}(\overline{\kappa}, \mathtt{ist}(\kappa_1, \phi)) \subseteq$ *Vocab*$(\mathfrak{M})$, and the $\mathrm{Dom}(\mathfrak{M})$ is a rooted subtree; i.e. if $\overline{\kappa} > \overline{\kappa}_0$, then not $\mathfrak{M}, \nu \models_{\overline{\kappa}} \chi$. We write $\mathfrak{M} \models_{\overline{\kappa}} \chi$ iff $(\mathrm{Vocab}(\overline{\kappa}, \chi) \subseteq$ *Vocab*$(\mathfrak{M})$ and $\forall \nu \in \overline{\kappa}^{\mathfrak{M}} \quad \mathfrak{M}, \nu \models_{\overline{\kappa}} \chi)$; we also write $\models_{\overline{\kappa}} \chi$ iff for all models $\mathfrak{M}$ classical on Vocab $\quad \mathfrak{M} \models_{\overline{\kappa}} \chi$.

3 Decidability

The purpose of this section is to show that the propositional logic of contexts is decidable (i.e. that there is an effective procedure that says whether or not a given formula is valid, and hence also a theorem of the system). This will be done by showing that the propositional logic of contexts has the finite model property: any formula that is satisfiable is satisfiable in a model with finitely many finite truth assignments.

Definition (restriction of $\mathfrak{M}(\overline{\kappa})$): We first define the restriction of a single truth assignment, ν, with respect to $\text{Vocab}(\overline{\kappa}_0, \phi)$ to be a truth assignment which, on the atoms that are in the scope of the context sequence $\overline{\kappa}$, corresponds to ν, and is false elsewhere.

$$\nu_{\text{Vocab}(\overline{\kappa}_0,\phi),\overline{\kappa}} = \text{ the unique } \nu' \text{ such that } \nu'(p) = \begin{cases} \nu(p) & <\overline{\kappa}, p> \in \text{Vocab}(\overline{\kappa}_0, \phi) \\ 0 & <\overline{\kappa}, p> \notin \text{Vocab}(\overline{\kappa}_0, \phi) \end{cases}$$

The restriction of a set of truth assignments, V, with respect to $\text{Vocab}(\overline{\kappa}_0, \phi)$ is defined as follows.

$$V_{\text{Vocab}(\overline{\kappa}_0,\phi)} = \{\nu_{\text{Vocab}(\overline{\kappa}_0,\phi),\overline{\kappa}} \,|\, \nu \in V\}.$$

Definition (restriction of $\mathfrak{M}$): The restriction of a model $\mathfrak{M}$ with respect to $\text{Vocab}(\overline{\kappa}_0, \phi)$ is a model which maps every context sequence which appears in ϕ to the set of restricted truth assignments.

$$\mathfrak{M}_{\text{Vocab}(\overline{\kappa}_0,\phi)} : S \to \{\mathfrak{M}(\overline{\kappa})_{\text{Vocab}(\overline{\kappa}_0,\phi)} \,|\, \overline{\kappa} \in \text{Dom}(\text{Vocab}(\overline{\kappa}_0, \phi))\},$$

where S is the set of all subsequences of context sequences from $\text{Dom}(\text{Vocab}(\overline{\kappa}_0, \phi))$.

$$S = \{\overline{\kappa}_1 \,|\, \overline{\kappa}_1 \leq \overline{\kappa}_0 \quad \wedge \quad \exists \overline{\kappa}_2 \in \text{Dom}(\text{Vocab}(\overline{\kappa}_0, \phi)) \quad \overline{\kappa}_2 \leq \overline{\kappa}_1\}.$$

Thus the model $\mathfrak{M}_{\text{Vocab}(\overline{\kappa}_0,\phi)}$ maps a context sequence $\overline{\kappa}$ to a set of truth assignments $\mathfrak{M}(\overline{\kappa})_{\text{Vocab}(\overline{\kappa}_0,\phi)}$:

$$\mathfrak{M}_{\text{Vocab}(\overline{\kappa}_0,\phi)} : \overline{\kappa} \mapsto \mathfrak{M}(\overline{\kappa})_{\text{Vocab}(\overline{\kappa}_0,\phi)}.$$

Theorem (finite model property): $\mathfrak{M} \models_{\overline{\kappa}_0} \phi$ iff $\mathfrak{M}_{\text{Vocab}(\overline{\kappa}_0,\phi)} \models_{\overline{\kappa}_0} \phi$.

Note that only the ($\Rightarrow$) direction will be needed to prove the decidability of the propositional logic of context. We actually prove a stronger property:

$$\text{if} \quad \text{Vocab}(\overline{\kappa}, \alpha) \subseteq \text{Vocab}(\overline{\kappa}_0, \phi) \quad \text{then} \quad (\mathfrak{M} \models_{\overline{\kappa}} \alpha \quad \text{iff} \quad \mathfrak{M}_{\text{Vocab}(\overline{\kappa}_0,\phi)} \models_{\overline{\kappa}} \alpha)$$

by induction on the structure of the formula ϕ.

Corollary (decidability): There is an effective procedure which will determine whether or not a formula given in some context is valid.

4 Comparison to Kripke Semantics

In this section we study the relationship between the semantics of context (as given in this paper) and Kripke semantics. We show that if all the aspects of partiality in the definition of a context model are disregarded, then most context models can be matched up to a particular class of Kripke models. Since there is always some leeway in the match, based on how we define the Kripke model and what notion of equivalence between context and Kripke models is taken, the models will not be matched precisely. We will discuss the adequacy of the match in more detail later.

We proceed to define some preliminaries for our construction.

Definition (non-partial model): A non-partial context model $\mathfrak{M}$ is a function which maps every context sequence $\overline{\kappa} \in \mathbb{K}^*$ to a set of total truth assignments,

$$\mathfrak{M} \in (\mathbb{K}^* \to \mathbf{P}(\mathbb{P} \to 2)).$$

Note that the two additional side conditions in the definition of the model are no longer needed. Also note that for the non-partial models the following property of the satisfaction relation holds:

$$\mathfrak{M}, \nu \models_{\overline{\kappa}} \neg\phi \quad \text{iff} \qquad \text{not} \quad \mathfrak{M}, \nu \models_{\overline{\kappa}} \phi.$$

Since in the comparison to Kripke semantics we will only be concerned with non-partial models, henceforth in this section we will refer to a partial context model simply as a context model. In this section we will also use the term "context logic" to refer to the general system described in §2 with the semantics restricted to non-partial models.

We now give a brief sketch of a standard propositional modal logic. We will be using a propositional modal logic with a countable number of modalities and its corresponding Kripke semantics. Given a context language specified by a possibly finite set of contexts, $\mathbb{K}$, and a set of propositional atoms, $\mathbb{P}$, we define a modal language consisting of the propositional atoms, $\mathbb{P}$, standard propositional connectives, $\neg$ and $\to$, and modalities, $\Box_1, \Box_2, \ldots$; one for each context from $\mathbb{K} = \{\kappa_\beta\}_{\beta<\alpha}$. We also define a bijective translation function which to each formula of the context logic, $\phi \in \mathbb{W}$, assigns a well-formed modal formula, $\phi^\Box$. The formula $\phi^\Box$ is obtained from ϕ by replacing each occurrence of $\mathtt{ist}(\kappa_\beta, \psi)$ in ϕ with $\Box_\beta(\psi^\Box)$. A *Kripke model* is a tuple $<\mathbb{S}, w_0, \pi, R_\beta>_{\beta<\alpha}$ where $\mathbb{S}$ is the set of *possible worlds*, $w_0 \in \mathbb{S}$ is the *actual world*, and π is a mapping from the the worlds in $\mathbb{S}$ to truth assignments over atomic propositions in $\mathbb{P}$, for some $\alpha \leq \omega$. Every R_β is a binary relation on $\mathbb{S}$. In order to distinguish Kripke models from context models, we use $\mathfrak{M}^\Box$ to refer to Kripke models. Kripke models are often called Kripke structures or possible-world structures. *Satisfaction* is defined to be a relation on a Kripke model, a world from that model, and a formula; it is written as $\mathfrak{M}^\Box, w \models \phi$. Note that the same symbol is used for satisfaction in the context logic, however it will be obvious from the arguments of the relation which satisfaction relation is being referred to. Atomic formulas are satisfied at a world if they are made true by the truth assignment associated with that world. Satisfaction for propositional connectives is defined as in classical propositional logic. The formula $\Box_\beta \, \phi$ is satisfied at a world w iff ϕ is satisfied at every world w' s.t. $w \, R_\beta \, w'$.

In order to compare a context model and a Krïpke model, we need to know which worlds are intended to describe the same state of affairs as a given context, i.e. which worlds are associated to which context.

Definition (association relation): A relation A is an association relation from the context model $\mathfrak{M}$ to the Kripke model $\mathfrak{M}^\Box = \langle \mathbb{S}, w_0, \pi, R_\beta \rangle_{\beta<\alpha}$ iff

1. $A \subseteq \mathbb{K}^* \times (\mathbb{P} \rightarrow 2) \times \mathbb{S}$
2. $(\forall \overline{\kappa})(\forall \nu \in \mathfrak{M}(\overline{\kappa}))(\exists w) \quad A(\overline{\kappa}, \nu, w)$
3. $A(\overline{\kappa}, \nu, w)$ implies $(\forall w')(\exists \beta)$ $(w R_\beta w'$ implies $(\exists \kappa')(\exists \nu')$ $A(\overline{\kappa} * \kappa', \nu', w'))$
4. $A(\overline{\kappa}, \nu, w)$ implies $\pi(w) = \nu$
5. $A(\epsilon, \pi(w_0), w_0)$

Given a context sequence $\overline{\kappa}$ and a truth assignment from $\mathfrak{M}(\overline{\kappa})$, the association relation expresses which world is to be associated with the truth assignment in that context, and vice versa. Note that the same truth assignment in different context sequences may produce different worlds.

In order to be able to compare a context model and a Kripke model, we need a notion of what it means for the two models to be equivalent.

Definition (elementary equivalence): A context model $\mathfrak{M}$ is elementarily equivalent to a Kripke model $\mathfrak{M}^\Box$ with respect to association A $(\mathfrak{M} \equiv \mathfrak{M}^\Box$ w.r.t. $A)$ iff

$$A(\overline{\kappa}, \nu, w) \quad \text{implies} \quad (\mathfrak{M}, \nu \models_{\overline{\kappa}} \phi \quad \text{iff} \quad \mathfrak{M}^\Box, w \models \phi^\Box)$$

for any context sequence $\overline{\kappa}$, truth assignment $\nu \in \mathfrak{M}(\overline{\kappa})$, world $w \in \mathbb{S}$, and wff ϕ.

We introduce a class of context models, called the *actual models.* They have the property that the empty context sequence is associated with a single truth assignment, which is interpreted as the actual state of affairs or the state of affairs in the actual world.

Definition ($\mathfrak{Actual}$): A context model $\mathfrak{M}$ is an *actual model*, $\mathfrak{M} \in \mathfrak{Actual}$, iff $|\mathfrak{M}(\epsilon)| = 1$.

Theorem (representation1): For every context model $\mathfrak{M} \in \mathfrak{Actual}$, there exists a Kripke model $\mathfrak{M}^\Box$, and an association A from $\mathfrak{M}$ to $\mathfrak{M}^\Box$ such that $\mathfrak{M} \equiv \mathfrak{M}^\Box$ w.r.t. A.

In the proof, given a context model $\mathfrak{M}$, we construct an elementarily equivalent Kripke model $\mathfrak{M}^\Box$. Intuitively, for every truth assignment in every context sequence, we create a world in the Kripke model. The function π of a world is the same as the truth assignment that created that world. We then define relations on these worlds and show that the created structure, $\mathfrak{M}^{\Box}$, is in fact a Kripke model. Finally, we prove that the two models are in fact elementarily equivalent.

We will not be able to represent all Kripke models using context models. To define the class of Kripke models which can be represented using context models, we introduce a new property of accessibility relations of Kripke structures.

Definition ($\Phi_1(R_i, R_j)$): The ordered pair of relations $<R_i, R_j>$ satisfies the property Φ_1, formally $\Phi_1(R_i, R_j)$, iff

$$(\forall w_1 \forall w_2 \forall w_2' \forall w_3)\ ((w_1 R_i w_2 \text{ and } w_2 R_j w_3 \text{ and } w_1 R_i w_2') \text{ implies } w_2' R_j w_3).$$

Definition (C_1): Let C_1 be the classes of Kripke models in which all the ordered pairs of accessibility relations satisfy the property Φ_1.

Theorem (representation2): For every Kripke model $\mathfrak{M}^{\Box} \in \mathsf{C}_1$ there exists a context model $\mathfrak{M}$ such that $\mathfrak{M} \equiv \mathfrak{M}^{\Box}$ w.r.t. A.

In the proof, given a Kripke model $\mathfrak{M}^{\Box}$, we construct an elementarily equivalent context model $\mathfrak{M}$. Intuitively, we will identify context sequences with paths through the Kripke model $\mathfrak{M}^{\Box}$. Then, the truth assignments of the worlds in $\mathfrak{M}^{\Box}$ which can be reached via a path $\overline{\kappa}$ will be placed in $\mathfrak{M}(\overline{\kappa})$. Finally, we prove that the two models are in fact elementarily equivalent.

4.1 Discussion

The representation results in this section depend on our definition of the Kripke model and the definition of elementary equivalence. Variations in either of these definitions would slightly change the theorems. For example, sometimes the actual world is excluded from the definition of the Kripke model. In this case we could generalize the (**representation1 theorem**) to hold for any context model, rather than only those in $\mathfrak{Actual}$. For (**representation2 theorem**) to hold we would need a way of including multiple subtrees in a single context model. One solution would be to connect all the subtrees to ϵ and associate the empty vocabulary with the empty sequence. To take another example, stronger results could be obtained by insisting that the association relation matches every world to some truth assignment in some context sequence. But here again, to prove (**representation2 theorem**) we would need vocabularies or a stronger notion of a context model which would allow multiple rooted subtrees as the domain of $\mathfrak{M}$. To conclude, there is always some leeway in representation results, based on the basic definitions. The main purpose of this section is to give the general flavor of the relations in expressiveness of the two kinds of models and a methodology which can be used for comparing context models and Kripke models.

5 Conclusions and Future Work

Our goal is to extend the system to a full quantification logic. One advantage of quantificational system is that it enables us to express relations between context, operations on contexts, and state *lifting rules* which describe how a fact from one context can be used in another context. However, in the presence of context variables it might not be possible to define the vocabulary of a sentence without knowing which object a variable is bound to. Therefore, the first step in this direction is to to examine propositional systems with dynamic definitions of meaningfulness.

We also plan to define non-Hilbert style formal systems for context. Probably the most relevant is a natural deduction system, which would be in line with McCarthy's original proposal of treating contextual reasoning as a strong version of natural deduction. In such a system, entering a context would correspond to making an assumption in natural deduction, while exiting a context corresponds to discharging an assumption.

5.1 Acknowledgements

The authors would like to thank Tom Costello, R. V. Guha, Furio Honsell, John McCarthy, Grigorii Mints, Carolyn Talcott and Johan F. A. K. van Benthem for their valuable comments.

This research is supported in part by the Advanced Research Projects Agency, ARPA Order 8607, monitored by NASA Ames Research Center under grant NAG 2-581, by NASA Ames Research Center under grant NCC 2-537, NSF grants CCR-8915663, and IRI-8904611 and Darpa contract NAG2-703.

References

1. Giuseppe Attardi and Maria Simi. A formalization of viewpoints. Technical Report TR-93-062, International Computer Science Institute, Berkeley, California 94704-1105, 1993.
2. D. A. Bochvar. Two papers on partial predicate calculus. Technical Report STAN-CS-280-72, Department of Computer Science, Stanford University, 1972. Translation of Bochvar's papers originally published in 1938 and 1943.
3. Saša Buvač, Vanja Buvač, and Ian A. Mason. Metamathematics of contexts. *Fundamenta Informaticae.* To appear.
4. Saša Buvač and Ian A. Mason. Propositional logic of context. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1993.
5. John McCarthy. Generality in artificial intelligence. *Comm. of ACM*, 30(12):1030–1035, 1987. Reprinted in [6].
6. John McCarthy. *Formalizing Common Sense: Papers by John McCarthy.* Ablex Publishing Corporation, 355 Chesnut Street, Norwood, NJ 07648, 1990.
7. John McCarthy. Notes on formalizing context. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993.

The Generalized Logic of only Knowing (GOL) that Covers the Notion of Epistemic Specifications

Jianhua Chen

Computer Science Department
Louisiana State University
Baton Rouge, LA 70803, USA
E-mail: jianhua@bit.csc.lsu.edu

Abstract

We define GOL, the generalized logic of only knowing, and propose to use it as a unified framework for non-monotonic reasoning. The GOL logic is a generalization of the logic of *only knowing* (OL) by Levesque [8], and it covers the important notion of *epistemic specification* by Gelfond [3, 5] which is very useful in knowledge representation. By giving a model-theoretic account for *epistemic specifications*, the GOL helps clarify the conceptual understanding for epistemic theories. The GOL logic contains the OL logic as a subset and thus it retains the important features of OL. The OL logic is a modal logic which can be used to formalize an agent's introspective reasoning and to answer epistemic queries to databases. In recent works [1-2], we established the relations between OL logic and MBNF (the logic of minimal belief and negation as failure by Lifschitz [9]), between OL and extended logic programs, and showed that OL is a fairly general logical framework for non-monotonic reasoning. This work further generalizes the OL logic to enhance the expressive power to include epistemic specifications. We also present a proof theory for GOL together with the soundness and completeness results. This work is restricted to the propositional case only.

1. INTRODUCTION

This work is the continuation of the efforts in [1-2] to establish the logic of "only knowing" (OL [8]) as a unified framework for non-monotonic reasoning. The main motivation is to extend the OL logic to include Gelfond's epistemic specifications (ES) [3, 5]. We propose the logic GOL which, in addition to the expressive power and inference capability of OL, covers the important notion of *epistemic specifications* and thus is a more powerful logic for knowledge representation. We give both the model-theoretic semantics of GOL and a proof theory of it, together with the soundness and completeness results.

The logic of only knowing is proposed by Levesque [8]. The motivation for the OL logic is to show that some patterns of non-monotonic reasoning can be captured by using the classical notions of logic (satisfiability, validity, logical consequence, etc). To achieve that, Levesque extends the classical modal logic by formalizing the notion of "only knowing" and introducing two modal operators "B" and "O" (only knowing) in the OL logic. In [8], a model-theoretic semantics is defined for OL and a simple proof theory is established. It is shown that the autoepistemic logic of Moore [13] can be embedded in OL.

Obviously, with the capability to express the notion of "only knowing", and a clear model-theoretic semantics, as well as a simple proof theory, the OL logic is a very attractive candidate to be a general framework for various forms of non-monotonic reasoning. A number of works have been devoted to clarifying the relationship between OL logic and other non-monotonic formalisms. A recent significant development of this aspect is the works [1, 10, 11] which established that extended logic programs [4] (with classical negation and epistemic disjunction) can be embedded in the AE-logic (hence in OL). In particular, our recent work [1-2] established that a substantial subclass of the MBNF logic, the logic of *minimal belief and negation as failure* by Lifschitz [9], can be embedded in OL. Since MBNF is shown to be a general framework for non-monotonic reasoning which covers various non-monotonic logics such as circumscription, default logic, the embedding of MBNF in OL shows that the OL logic can also be used as a general logic framework for non-monotonic reasoning. Some recent works [6, 7] developed the multiple-agent version of the only knowing logic.

Nice as the OL logic is to be a general non-monotonic reasoning framework, it has a limitation in expressive power: it does not cover the important notion of *epistemic specifications* [3, 5] (called *strong introspection* in [3]) of Gelfond which is very useful for knowledge representation. The main purpose of *epistemic specifications* (ES) is to further expand the extended logic programs in order to allow for correct representation of incomplete information in the presence of multiple extensions. The idea is to include a *stronger* notion of introspection in the language by adding modal operators **K** and **M**, whose meaning depends on not only the current set of beliefs, but also all sets of beliefs of an agent. Gelfond defined the syntax and *world view* semantics for epistemic specifications for the first-order case, and also generalized the work to allow the absence of *domain closure assumption*. Many interesting applications of epistemic theories to knowledge representation have been illustrated [3, 5].

Although the OL logic cannot cover the epistemic specifications, a closer look does reveal a significant resemblance between the notion of "only knowing" in OL and the the notion of "world view" in epistemic theories. Essentially, a set of propositional interpretations W is said to satisfy $\mathbf{O}\phi$ (where ϕ is a propositional formula) if W is precisely the set of interpretations satisfying ϕ. On the other hand, a collection A of sets of propositional literals is a world view for an epistemic theory Δ, if A is precisely the set of all belief sets of Δ with respect to A. The ideas behind "only knowing" and "world view" are thus very much similar. Therefore, it is possible to extend the OL logic in order to give a model-theoretic account of epistemic specifications. This is what we attempt to accomplish in this paper.

The main benefits of our approach are the following. First, by defining GOL and relating GOL to the notion of epistemic specifications, we are able to give a clear, declarative semantics for epistemic specifications. This is similar to the original motivation of Levesque [8] in proposing the OL logic, namely, to capture non-monotonic reasoning in a generalized framework of classical logic, using the traditional notions of validity, satisfiability, etc. Second, the GOL logic proposed can be used as a general framework for formalizing various non-monotonic reasoning, given the facts that GOL covers epistemic specifications and that the OL logic is a subset of GOL. This will offer a unified framework to compare various non-monotonic formalisms based on the same application domain. Finally, the proof theory of GOL defined in this paper can be usefully adopted to perform query-answering for epistemic specifications which so far do not have well-established query-answering procedures.

This paper is organized as follows. In section 2, we will give basic definitions, briefly review OL and the notion of epistemic specifications. In Section 3, we define the logic GOL, and in Section 4, we establish its relation with epistemic specifications. The proof theory for GOL together with the soundness and completeness results will be presented in Section 5. We conclude in Section 6. For the sake of simplicity, throughout the paper we restrict the focus to *propositional* modal logics only, and leave the generalization to first-order case as future work.

2. DEFINITIONS

We give the basic definitions and briefly review OL and ES in this section. The reader is referred to [3, 5, 8] for detailed discussion about these two logics. In the propositional fragment of OL and ES, we will deal with propositional languages extended by adding some modal operators ("B" and "O" in OL, "K" and "M" in ES). The propositional language contains possibly infinite number of proposition symbols and ordinary logical connectives $\wedge$, $\neg$. An OL theory is a finite set of formulas and an epistemic theory is a finite set of epistemic specifications (to be defined later). A formula is called *objective* if it does not include any modal operators; it is called *subjective* if each occurrence of objective formula is within the scope of a modal operator. Objective theories and subjective theories are defined similarly. A proposition p_j is also called an *atom*. A *literal* is either an atom or the negation of an atom. An *interpretation* I is a set of atoms which are assigned the truth value "true" in the interpretation. We use Ω to denote the set of all interpretations, and *Lit* to denote the set of all literals in the language. For an objective theory Φ, Mod(Φ) denotes the set of propositional models of Φ.

2.1 The logic of Only Knowing

Levesque developed [8] the logic of *only knowing* (OL) to formulate and infer about an agent's knowledge and belief. An OL language is obtained by adding two modal operators **B** and O to a propositional language. Formulas are formed in the obvious way, and there is no restriction as to the level of nestings of the modal operators. An OL theory is called *basic* if it does not contain any occurrence of the O operator.

The structures which define the truth/falsity of a formula ϕ in the OL language will be different from those used in defining the truth value of ordinary propositional formulas. Consider a theory Δ in OL, with the set of all interpretations Ω. A *bimodal structure* is of the form $\langle I, S\rangle$ where $I \in \Omega$ is an interpretation and $S \subseteq \Omega$ is a set of interpretations. Intuitively, I represents the "real world" and S represents the set of "possible worlds" accessible from I. Notice that I need not be a member of S. Essentially, the truth value of an OL formula ϕ will be determined at each structure $\langle I, S\rangle$.

Given a structure $\langle I, S\rangle$ and formulas ϕ and ψ in OL, the following relation "$\models_{ol}$" specifies the rules for truth value assignment:

(1) $\langle I, S\rangle \models_{ol} \phi \Leftrightarrow \phi \in I$ for an atom ϕ.

(2) $\langle I, S\rangle \models_{ol} \neg\phi \Leftrightarrow \langle I, S\rangle \not\models_{ol} \phi$.

(3) $\langle I, S\rangle \models_{ol} \phi \wedge \psi \Leftrightarrow \langle I, S\rangle \models_{ol} \phi$ and $\langle I, S\rangle \models_{ol} \psi$.

(4) $\langle I, S\rangle \models_{ol} \mathbf{B}\phi \Leftrightarrow$ for every $J \in S$, $\langle J, S\rangle \models_{ol} \phi$.

(5) $\langle I, S\rangle \models_{ol} \mathbf{O}\phi \Leftrightarrow \langle I, S\rangle \models_{ol} \mathbf{B}\phi$ and for any $J \in \Omega$, $\langle J, S\rangle \models_{ol} \phi \Rightarrow J \in S$.

It is clear that for a structure $\langle I, S\rangle$, the truth value of an objective formula does not depend on S and the truth value of a subjective formula does not depend on I. We say $\langle I, S\rangle$ satisfies a formula ϕ if $\langle I, S\rangle \models_{ol} \phi$. In order to define the notion of a *model* (and hence the *satisfiability*) for an OL theory Δ, we need to take care of an additional technical detail. We say that two sets of interpretations S_1 and S_2 are *equivalent* if we have Belief(S_1) = Belief(S_2), where Belief(S) = $\{\alpha : \alpha$ is basic and $\langle I, S\rangle \models_{ol} \mathbf{B}\alpha\}$. A set S of interpretations is said to be *maximal* (or we say the structure $\langle I, S\rangle$ is maximal) if S is not equivalent to any of its proper superset. A *model* of a theory Δ is a maximal structure $\langle I, S\rangle$ such that $\langle I, S\rangle \models_{ol} \phi$ for each $\phi \in \Delta$. A theory is *satisfiable* if it has a model. A theory Δ entails a formula ϕ ($\Delta \models_{ol} \phi$) if $\Delta \cup \{\neg\phi\}$ is not satisfiable.

2.2 Epistemic Specifications

Gelfond proposed the notion of *epistemic specifications* [3] and then generalized it to allow the absence of the domain closure assumption [5]. The following presentation about the propositional fragment of ES theories conforms to the formulation in [5]. The language L_0 for propositional epistemic theories is obtained by adding two modal operators "**K**" and "**M**" to a propositional language, with propositions p, q, ..., logical connectives $\wedge$, $\neg$, and a Boolean constant **true**. An *epistemic specification* is a finite collection of rules of the form

$$\phi \leftarrow \phi_1, \ldots, \phi_m, \phi_{m+1}, \ldots, \phi_n, not\phi_{n+1}, \ldots, not\phi_k,$$

where $\phi_{m+1} \ldots \phi_n$ are subjective, while ϕ and other ϕ_j's are objective, *not* denotes negation as failure.

In this logic, the structure used to define the true/false values of formulas are of the form $\langle W, A\rangle$, where W is a set of literals and A is a collection of such sets. Intuitively, W can be understood as a reasoner's current belief set, while A can be viewed as a collection of possible belief sets of the reasoner. The notion of truth ($\models$) and falsity ($=\!|$) of any ES formula with respect to $\langle W, A\rangle$ is inductively defined as follows:

$\langle W, A\rangle \models \mathrm{p} \Leftrightarrow \mathrm{p} \in W$ for an atom p.

$\langle W, A\rangle \models \mathbf{K}\phi \Leftrightarrow$ for each $A_j \in A$, $\langle A_j, A\rangle \models \phi$.

$\langle W, A\rangle \models \mathbf{M}\phi \Leftrightarrow$ for some $A_j \in A$, $\langle A_j, A\rangle \models \phi$.

$\langle W, A\rangle \models \phi \wedge \psi \Leftrightarrow \langle W, A\rangle \models \phi$ and $\langle W, A\rangle \models \psi$.

$\langle W, A\rangle \models \neg\phi \Leftrightarrow \langle W, A\rangle =\!| \phi$.

$\langle W, A\rangle =\!| \mathrm{p} \Leftrightarrow \neg\mathrm{p} \in W$ for an atom p.

$\langle W, A\rangle =\!| \mathbf{K}\phi \Leftrightarrow \langle W, A\rangle \not\models \mathbf{K}\phi$.

$\langle W, A\rangle =\!| \mathbf{M}\phi \Leftrightarrow \langle W, A\rangle \not\models \mathbf{M}\phi$.

$\langle W, A\rangle =\!| \phi \wedge \psi \Leftrightarrow \langle W, A\rangle =\!| \phi$ or $\langle W, A\rangle =\!| \psi$.

$\langle W, A\rangle =\!| \neg\phi \Leftrightarrow \langle W, A\rangle \models \phi$.

Another connective *or* is also defined for the language, ϕ *or* ψ if and only if $\neg(\neg\phi \wedge \neg\psi)$. The introduction of *or* is mainly for simplicity of expressions.

The notions of a *belief set* and *world-view* play a crucial role. Let Δ be an epistemic specification, let W be a set of literals and A be a collection of such sets. We say that W is a belief set of Δ with respect to A if W satisfies certain conditions. For an objective epistemic specification Δ *without negation as failure*, W is a belief set of Δ (with respect to any A) if W is a *minimal* set of literals satisfying the following conditions:

1. For each rule $\phi \leftarrow \phi_1, \phi_2, ..., \phi_m$ from Δ, if $W \models \phi_1 \wedge ... \wedge \phi_m$, then $W \models \phi$;
2. If W contains complement literals, then $W = Lit$.

Now let Δ be an objective epistemic specification and W be a set of literals. Define the epistemic specification Δ_W to be the set of rules obtained by the following operations.

1. Remove from Δ the rules with formula $not\phi_j$ in the premise such that $W \models \phi_j$;
2. Remove from the premise of the remaining rules any formulas of the form $not\phi_j$.

W is a belief set of Δ if and only if W is a belief set of Δ_W.

Finally, let Δ be an arbitrary epistemic specification. Let W be a set of literals and let A be a collection of such sets. Define Δ_A, the reduction of Δ with respect to A as the set of rules obtained by the following operations.

1. Remove from Δ all rules whose premise contains a subjective formula ϕ_j such that $A \not\models \phi_j$;
2. Remove from the premise of the remaining rules any subjective formulas.

W is a belief set of Δ with respect to A if and only if W is a belief set of Δ_A. Note that when Δ is objective, Δ_A is the same as Δ. We use $\langle W, A\rangle \models_{es} \Delta$ to denote " W is a belief set of Δ w.r.t. A".

Let A be a non-empty collection of sets of literals. We say that A is a *world view* of Δ if and only if A is precisely the set of all *consistent* ($\neq Lit$) belief sets of Δ with respect to A, i.e., A satisfies the fix-point equation

$$A = \{W\colon \langle W, A\rangle \models_{es} \Delta \text{ and } W \neq Lit\}.$$

Intuitively, the idea is that a collection A of sets of literals is accepted as a world view only when each belief set W ($\in A$) assigns truth values exactly according to the global situation in A, and A is a *maximal* collection of such W's. Gelfond has shown how to use epistemic specifications to represent incomplete knowledge, to model the closed-world assumption, integrity constraints in deductive databases, etc.

3. THE DEFINITION OF GOL

3.1. A Motivating Example

Example 1. As we have discussed briefly in the introduction section, there is a close resemblance between the notion of "only knowing" in OL and the notion of "world view" in ES. Consider the following ES theory Δ which is essentially from [3]:

Eligible ← HighGPA.
Eligible ← Minority, FairGPA.
¬Eligible ← ¬FairGPA.
Interview ← ¬**K**Eligible, ¬**K**¬Eligible.
HighGPA *or* FairGPA

It is easy to see that Δ has exactly one world view A = {{HighGPA, Eligible, Interview}, {FairGPA, Interview}}. The definition of world view tells us that the set A satisfies the fixed-point equation $A = \{W\text{: } \langle W, A\rangle \models_{es} \Delta\}$. On the other hand, let $S \subseteq \Omega$ be a set of interpretations and consider an OL theory Φ. $\langle I, S\rangle \models_{ol} \mathbf{O}\Phi$ if $S = \{I\ : \langle I, S\rangle \models_{ol} \Phi\}$. We can observe the similarity between these two fixed-point definitions. This suggests that we can possibly generalize the notion of "only knowing" to model the "world view" in ES. The generalization lies in using structures of the form $\langle W', A'\rangle$ in order to model "$\langle W, A\rangle \models_{es} \Delta$" in the generalized logic, where $W' = \text{Mod}(W)$, $A' = \{\text{Mod}(A_j)\text{: } A_j \in A\}$. Note that in ES, the definition for "$\langle W, A\rangle \models_{es} \Delta$" is in fact given by a fix-point operation rather than declaratively. By using the structures like $\langle W', A'\rangle$, we can declaratively define the notion "W is a belief set of Δ w.r.t. to A" as $\langle W', A'\rangle \models_{gol} \Delta'$ where Δ' the counter-part of Δ in the new (GOL) logic. We can subsequently formalize the notion $A = \{W\text{: } \langle W, A\rangle \models_{es} \Delta\}$ as follows: A satisfies the fixed-point equation if and only if $A' = \{W'\text{: } \langle W', A'\rangle \models_{gol} \Delta'\}$. Essentially, this means to define a new "only knowing" operator $\mathbf{O}_2$ (and also a new "belief" operator $\mathbf{B}_2$) which characterizes collections A' of sets of interpretations, i.e., $\langle W', A'\rangle \models_{gol} \mathbf{O}_2\Delta'$ if and only if $A' = \{W'\text{: } \langle W', A'\rangle \models_{gol} \Delta'\}$. For this particular theory Δ, A' = {Mod({HighGPA, Eligible, Interview}), Mod({FairGPA, Interview})} is the only (GOL) model for $\mathbf{O}_2\Delta'$ (we will show later how Δ' looks). ♣

3.2 Defining the GOL

From the analysis in the above example, we see that the new operators $\mathbf{B}_2$, $\mathbf{O}_2$ in GOL should be a generalization of the original **B**, **O** operators in OL, because the $\mathbf{B}_2$, $\mathbf{O}_2$ operators are applied to a collection of sets of interpretations whereas the **B**, **O** operators are applied to a set of interpretations. This suggests to extend the language of OL by adding the modal operators $\mathbf{B}_2$ and $\mathbf{O}_2$ to obtain the language of GOL, and to use structures of the form $\langle I, S, V\rangle$ in defining the semantics of GOL, where I is an interpretation, S is a set of interpretations and V is a collection of sets of interpretations.

We consider a modal language which is obtained by adding the following four modal operators "$\mathbf{B}_1$", "$\mathbf{O}_1$", "$\mathbf{B}_2$" and "$\mathbf{O}_2$" to a propositional language. We also include the propositional constants "*true*", "*false*" in the language. The GOL formulas are then defined to be any well-formed formulas which satisfy the additional requirement that the "$\mathbf{B}_2$", "$\mathbf{O}_2$" operators are applied only to *subjective* formulas. Therefore we do not allow any formulas of the form $\mathbf{B}_2\phi$, $\mathbf{O}_2\phi$ where ϕ is objective. Here the "$\mathbf{B}_1$", "$\mathbf{O}_1$" operators are precisely the "**B**", "**O**" operators in OL and hence the OL logic is a sub-logic of GOL. The truth value for formulas which do not involve the operators "$\mathbf{B}_2$" and "$\mathbf{O}_2$" is defined (as before) on the bimodal structure $\langle I, S\rangle$, whereas the truth value for general GOL formulas should be defined on the extended structure (a triple) of the form $\langle I, S, V\rangle$. We use "$\wedge$" and "$\neg$" as basic connectives, and others (such as "$\vee$" and "$\rightarrow$") are introduced as abbreviations of formulas in "$\wedge$" and "$\neg$".

The truth value definition for the formulas is given below:

(1) $\langle I, S, V\rangle \models_{gol} \phi \Leftrightarrow I \models \phi$ for objective ϕ.

(2) $\langle I, S, V\rangle \models_{gol} \neg\phi \Leftrightarrow \langle I, S, V\rangle \not\models_{gol} \phi$.

(3) $\langle I, S, V\rangle \models_{gol} \phi \wedge \psi \Leftrightarrow \langle I, S, V\rangle \models_{gol} \phi$ and $\langle I, S, V\rangle \models_{gol} \psi$.

(4) $\langle I, S, V\rangle \models_{gol} \mathbf{B}_1\phi \Leftrightarrow \langle J, S, V\rangle \models_{gol} \phi$ for each $J \in$ S.

(5) $\langle I, S, V\rangle \models_{gol} \mathbf{O}_1\phi \Leftrightarrow \langle I, S, V\rangle \models_{gol} \mathbf{B}_1\phi$ and $\langle J, S, V\rangle \models_{gol} \phi$ implies $J \in S$.

(6) $\langle I, S, V\rangle \models_{gol} \mathbf{B}_2\phi \Leftrightarrow \langle I, W, V\rangle \models_{gol} \phi$ for each $W \in V$.

(7) $\langle I, S, V\rangle \models_{gol} \mathbf{O}_2\phi \Leftrightarrow \langle I, S, V\rangle \models_{gol} \mathbf{B}_2\phi$ and $\langle I, W, V\rangle \models_{gol} \phi$ implies $W \in V$ for any $W \subseteq \Omega$.

Given the above truth value definitions, we can now proceed to define the notion of a *model* of a GOL theory Δ. Recall the notion of a *maximal* set of interpretations - S is maximal if S is not equivalent to any of its proper super sets. We now term such maximality as S-maximal. Given V, a collection of sets of interpretations, we define Belief2(V) = $\{\phi : \phi$ does not contain the $\mathbf{O}_2$ operator and $V \models_{gol} \mathbf{B}_2\phi\}$. Two collections of sets of interpretations V_1 and V_2 are said to be V-equivalent if Belief2(V_1) = Belief2(V_2). Define a collection V of sets of interpretations to be $V-$ *maximal* if V is not V-equivalent to any of its supersets. We now define a triple $\langle I, S, V\rangle$ to be *maximal* if and only if the following 3 conditions hold simultaneously: (1). S is S-maximal; (2). Each $S' \in V$ is S-maximal; (3). V is V-maximal.

A maximal triple $\langle I, S, V\rangle$ is a model of a GOL theory Δ if and only if $\langle I, S, V\rangle \models_{gol} \phi$ for each $\phi \in \Delta$. A theory is satisfiable if it has a model. A theory Δ entails a formula ϕ if $\Delta \cup \{\neg\phi\}$ is not satisfiable. A formula is valid if it is entailed by the empty theory. Obviously, the above definition degenerates to the OL logic definition when the theory Δ does not contain the operators "$\mathbf{B}_2$" and "$\mathbf{O}_2$".

4. THE GOL COVERS EPISTEMIC SPECIFICATIONS

We develop the relationship between the GOL and the notion of epistemic specifications of Gelfond. First we develop some syntactical properties of the epistemic specifications. Define formulas of the form $\mathbf{M}\phi$, $\mathbf{K}\phi$ (ϕ is any formula) and their negations as *belief literals*. A formula of the form (ϕ_1 *or* ϕ_2 *or* $\cdots$ *or* ϕ_n) is called a *clause*, where each ϕ_j is either a literal or a belief literal.

Lemma 1. Each objective formula ϕ can be expressed in its *conjunctive normal form* as $\phi = \bigwedge_{i=1}^{n} C_i$, where each C_i is an objective clause.

Lemma 2. Each formula ϕ in the language L_0 for ES (without "$\rightarrow$") can be expressed by its *conjunctive normal form* as

$$\phi = \bigwedge_{i=1}^{n} (\phi_i \text{ or } \sigma_i)$$

where each ϕ_i is an objective clause, each σ_i is a subjective clause without nested modal operators, and each objective formula ψ within a modal operator is in conjunctive normal form.

Definition. (mapping from epistemic specifications to GOL formulas). We define two mappings π_1 and π_2, where π_1 is applicable only to objective formulas. Here we assume that each formula is expressed by its *conjunctive normal form* as guaranteed by Lemma 2.

π_1	p_j	→	$p_j \wedge \mathbf{B}_1 p_j$	p_j is an atom
	$\neg p_j$	→	$\neg p_j \wedge \mathbf{B}_1(\neg p_j)$	p_j is an atom
	$\phi\ or\ \psi$	→	$\pi_1(\phi) \vee \pi_1(\psi)$	
	$\phi \wedge \psi$	→	$\pi_1(\phi) \wedge \pi_1(\psi)$	
π_2	p_j	→	$\mathbf{B}_1 p_j$	p_j is an atom
	$\neg p_j$	→	$\mathbf{B}_1(\neg p_j)$	p_j is an atom
	$\phi\ or\ \psi$	→	$\pi_2(\phi) \vee \pi_2(\psi)$	
	$\phi \wedge \psi$	→	$\pi_2(\phi) \wedge \pi_2(\psi)$	
	$\mathbf{K}\phi$	→	$\mathbf{B}_2\pi_2(\phi)$	ϕ is objective
	$\mathbf{M}\phi$	→	$\neg\mathbf{B}_2\neg\pi_2(\phi)$	ϕ is objective
	$\neg\mathbf{K}\phi$	→	$\neg\mathbf{B}_2\pi_2(\phi)$	ϕ is objective
	$\neg\mathbf{M}\phi$	→	$\mathbf{B}_2\neg\pi_2(\phi)$	ϕ is objective
	$not\phi$	→	$\neg\pi_2(\phi)$	ϕ is objective

Given an epistemic specification rule of the form

$$\mathrm{r}\ :\ \phi \leftarrow \phi_1, ..., \phi_m, \phi_{m+1}, ..., \phi_n, not\phi_{n+1}, ..., not\phi_k,$$

with each formula ϕ_j and ϕ in conjunctive normal form, we define the GOL formula $\pi(\mathrm{r})$ to be

$$\pi(\mathrm{r})\ :\ \pi_1(\phi) \leftarrow \pi_1(\phi_1), ..., \pi_1(\phi_m), \pi_2(\phi_{m+1}), ..., \pi_2(\phi_n), \pi_2(not\phi_{n+1}), ..., \pi_2(not\phi_k).$$

Note that here the connective "←" is the classical implication whereas in ES, "←" is interpreted constructively. Let Δ be an epistemic specification. Define the associated GOL theory $\Delta_1 = \{\pi(\mathrm{r}) \colon \mathrm{r} \in \Delta\}$, define $\Delta' = \mathbf{O}_1\Delta_1 \wedge \neg\mathbf{B}_1 false$. ♣

Example 2. Consider the epistemic specification Δ in Example 1. Applying the above definition, we have Δ_1 =

$$\begin{array}{lll}
\{\text{Eligible} \wedge \mathbf{B}_1\text{Eligible} & \leftarrow & \text{HighGPA} \wedge \mathbf{B}_1\text{HighGPA}, \\
\text{Eligible} \wedge \mathbf{B}_1\text{Eligible} & \leftarrow & \text{Minority} \wedge \mathbf{B}_1\text{Minority} \wedge \text{FairGPA} \wedge \mathbf{B}_1\text{FairGPA}, \\
\neg\text{Eligible} \wedge \mathbf{B}_1\neg\text{Eligible} & \leftarrow & \neg\text{FairGPA} \wedge \mathbf{B}_1\neg\text{FairGPA}, \\
\text{Interview} \wedge \mathbf{B}_1\text{Interview} & \leftarrow & \neg\mathbf{B}_2\mathbf{B}_1\text{Eligible} \wedge \neg\mathbf{B}_2\mathbf{B}_1\neg\text{Eligible}, \\
\multicolumn{3}{l}{\text{HighGPA} \wedge \mathbf{B}_1\text{HighGPA} \vee \text{FairGPA} \wedge \mathbf{B}_1\text{FairGPA}\},}
\end{array}$$

and $\Delta' = \mathbf{O}_1\Delta_1 \wedge \neg\mathbf{B}_1 false$.

Now we are ready to explore the connections between the GOL and ES. Given any set W of literals, the associated set S of interpretations is uniquely defined as $S = \mathrm{Mod}(W)$. Similarly, for any given collection A of sets of literals, the associated collection of sets of interpretations is $V_A = \{\mathrm{Mod}(W) \colon W \in A\}$. The main theorem is the following:

Theorem 1. Let Δ be an epistemic specification in conjunctive normal form. Then a collection A of sets of literals is a world view of Δ if and only if the associated V_A is a model of the formula $\mathbf{O}_2\Delta' \wedge \neg\mathbf{B}_2(\mathbf{B}_1 false)$.

5. A PROOF THEORY FOR THE GOL LOGIC

We present a simple proof theory for GOL which is very much of the same flavor as the proof theory for OL defined by Levesque [8]. The proof theory is both sound and complete.

First we list the proof theory for OL, since the OL logic is a subset of GOL. The proof theory for the OL logic introduces another modal operator "N" which is defined as follows: $\langle I, S\rangle \models_{ol} \mathbf{N}\phi$ if and only if for any $J \notin S$, $\langle J, S\rangle \models_{ol} \phi$. Consequently, $\mathbf{O}\phi$ is defined as $\mathbf{B}\phi \wedge \mathbf{N}\neg\phi$. The OL proof theory, which is sound and complete for the propositional OL, consists of the following axiom schemata and inference rules [8]:

(1) Propositional basis:

- All valid formulas of propositional logic.
- From α and $(\alpha \rightarrow \beta)$, infer β.

(2) Weak S5 for the operators **B** and **N**:

- $\mathbf{B}\phi$, where ϕ is a valid propositional formula.
- $\mathbf{B}(\phi \rightarrow \psi) \rightarrow (\mathbf{B}\phi \rightarrow \mathbf{B}\psi)$.
- $\sigma \rightarrow \mathbf{B}\sigma$ for any subjective σ.
  and the equivalent axioms for **N**;

(3) **N** vs. **B** axiom:
$\mathbf{N}\phi \rightarrow \neg\mathbf{B}\phi$, where ϕ is a falsifiable objective formula.

(4) $\mathbf{O}\phi \equiv \mathbf{B}\phi \wedge \mathbf{N}\neg\phi$.

Since the OL logic is a sub-logic of GOL, we certainly will have the above axiom schemata and inference rules (1) - (4) in the proof theory for GOL on replacing "**B**" with "$\mathbf{B}_1$" and "**N**" with "$\mathbf{N}_1$". We obviously need axiom schemata for formulas involving the modal operators $\mathbf{B}_2$ and $\mathbf{O}_2$. Similar to the OL proof theory, we introduce the modal operator $\mathbf{N}_2$ as follows. A structure $\langle I, S, V\rangle$ satisfies $\mathbf{N}_2\phi$ if and only if for any $W \subseteq \Omega$ and $W \notin V$, $\langle I, W, V\rangle \models_{gol} \phi$. Subsequently, we define $\mathbf{O}_2\phi \equiv \mathbf{B}_2\phi \wedge \mathbf{N}_2\neg\phi$. Other axiom schemata in GOL are defined as follows.

(a) $\mathbf{B}_2\phi$, $\mathbf{N}_2\phi$, where ϕ is a valid subjective OL formula.

(b) $\mathbf{B}_2(\phi \rightarrow \psi) \rightarrow (\mathbf{B}_2\phi \rightarrow \mathbf{B}_2\psi)$.

(c) $\mathbf{N}_2(\phi \rightarrow \psi) \rightarrow (\mathbf{N}_2\phi \rightarrow \mathbf{N}_2\psi)$.

(d) $\sigma \rightarrow \mathbf{B}_2\sigma$, $\sigma \rightarrow \mathbf{N}_2\sigma$, where σ is subjective and each occurrence of the operators $\mathbf{B}_1$ and $\mathbf{N}_1$ is in the scope of either $\mathbf{B}_2$ or $\mathbf{N}_2$.

(e) $\mathbf{B}_2\phi \rightarrow \neg\mathbf{N}_2\phi$, where ϕ is a falsifiable subjective OL formula (free of operators $\mathbf{B}_2$ and $\mathbf{N}_2$).

Note that we restrict the language of GOL in such a way that for each formula ϕ in the language, any objective sub-formula ψ will not appear directly within the scope of the operators $\mathbf{B}_2$ and $\mathbf{O}_2$.

Theorem 2. The proof theory for GOL is both sound and complete.

6. CONCLUSIONS

In this paper, we introduce the logic GOL as a generalization of the logic of only knowing. The objective is to enhance the expressive power of the OL logic to cover the notion of *epistemic specifications* and to obtain a general framework for non-monotonic reasoning. The resulting GOL logic is very expressive and powerful, in the sense that it includes both OL and *epistemic specifications*, and that various major non-monotonic formalisms, such as circumscription, extended logic programs, a large subset of default logic and the logic MBNF, can be embedded in GOL. The work here on GOL in fact relates to the works in [6, 7] on multi-agent logic of only knowing. In summary, GOL can be used as a general framework for non-monotonic reasoning in which one can compare various formalisms for applications in the same domain. The GOL provides a model-theoretic account for epistemic specifications and thus helps clarify the conceptual understanding of epistemic theories. We also give a sound and complete proof theory for GOL, which can be usefully adopted to perform query-answering for epistemic theories.

7. REFERENCES

[1] J. Chen, Minimal Knowledge + Negation as Failure = Only Knowing (sometimes), *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning*, June 1993, pp. 132-150.

[2] J. Chen, The Logic of Only Knowing as a Unified Framework for Non-monotonic Reasoning, To appear in the *Fundamenta Informatica* journal.

[3] M. Gelfond, Strong Introspection, *Proceedings of AAAI*, 1991, pp. 386-391.

[4] M. Gelfond and V. Lifschitz, Classical Negation in Logic Programs and Disjunctive Databases, *New Generation Computing*, 1991, To appear.

[5] M. Gelfond and H. Przymusinska, Reasoning in Open Domains, *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning*, June 1993, pp. 397-413.

[6] J. Halpern, Reasoning about Only Knowing with Many Agents, *Proceedings of National Conference on AI (AAAI'93)*, July 1993, pp. 655-661.

[7] G. Lakemeyer, All They Know: A Study in Multi-Agent Autoepistemic Reasoning, *Proceedings of the 13th International Joint Conference on AI*, 1993, pp. 376-381.

[8] H. J. Levesque, All I Know: A Study in Autoepistemic Logic, *Artificial Intelligence*, **42** (1-2) 1990, pp. 263-309.

[9] V. Lifschitz, Minimal Belief and Negation As Failure, Submitted, 1992.

[10] V. Lifschitz, G. Schwarz, Extended Logic Programs as Autoepistemic Theories, *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning*, June 1993, pp. 101-114.

[11] W. Marek and M. Truszczynski, Reflexive Autoepistemic Logic and Logic Programming, *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning*, June 1993, pp. 115-131.

[12] R. Moore, Semantical Considerations on Nonmonotonic Logic, *Artificial Intelligence*, **25** (1), 1985, pp. 75-94.

Reasoning about the safety of information : from logical formalization to operational definition

Laurence Cholvy[1] and Robert Demolombe[1] and Andrew Jones[2]

[1] ONERA/CERT, Toulouse, France
[2] Department of Philosophy and Norwegian Research Centre for Computers and Law, University of Oslo, Norway

Abstract. We assume that safety of information stored in a database depends on the reliability of the agents who have performed the insertions in the database. We present a logic S to represent information safety, and to derive answers to standard queries and to safety queries. The design of this logic is based on signaling act theory. Two strong simplifications lead to a logic S" with two modalities to represent explicit beliefs and implicit beliefs. Then, we present an operational view of S" in terms of First Order Logic, with meta predicates, which is implemented by a Prolog meta program. It is proved that answers derived in S" and computed by the meta program are identical. This property gives a clear meaning to computed answers.

keywords: Logic for Artificial Intelligence, Modal Logic, Reliability.

1 Introduction

The content of a database is usually considered as a set of database beliefs, and the only part which is recognized to represent true beliefs is the set of integrity constraints [10, 3]. However there are many situations where we can have a more refined information about database safety. Indeed, we may know for some particular sentences that they represent true beliefs if they are inserted by reliable agents. In that case the derivation of answers to standard queries may involve both beliefs and true beliefs, and it is not easy to know what is the status of the answer. In this context safety queries are defined to determine the part of the standard answers that represent true beliefs. Such a situation is illustrated by the following small example.

Let a, b, and c be three agents each capable of inserting information in a database. Agent a is an expert in economics and he knows some empirical rules which hold during particular periods of time. In particular a is an expert judge of the respective circumstances (not formally specifiable) under which the following two rules hold: "If taxes increase then unemployment increases", and, "If taxes increase then unemployment does not increase". If the propositions "taxes increase" and "unemployment increases" are respectively denoted by A and B, then the formal representation of these two rules is: $A \rightarrow B$ and $A \rightarrow \neg B$.

The fact that the system which manages the database knows that agent a is an expert for these two rules means that if, at a given time, a asserts $A \rightarrow B$

(resp. $A \rightarrow \neg B$), that is, if he inserts the rule $A \rightarrow B$ (resp. $A \rightarrow \neg B$) in the database, then $A \rightarrow B$ (resp. $A \rightarrow \neg B$) is guaranteed to be true. This fact is denoted by $\text{Safe}(a, A \rightarrow B)$ (resp. $\text{Safe}(a, A \rightarrow \neg B)$). It is important to notice that the system knows $\text{Safe}(a, A \rightarrow B)$ and $\text{Safe}(a, A \rightarrow \neg B)$ independently of the fact that agent a has inserted, or not, one of the two rules $A \rightarrow B$ or $A \rightarrow \neg B$.

Agent b is an expert in sociology and he knows that under some circumstances the following rule holds: "If unemployement increases and social rights are restricted, then criminality increases". If propositions "social rights are restricted" and "criminality increases" are respectively denoted by C and D, the rule is represented by: $B \wedge C \rightarrow D$, and reliability of b in regard to this rule is represented by: $\text{Safe}(b, B \wedge C \rightarrow D)$.

Finally agent c is a representative of the governement and he knows whether taxes increase, or not, and he knows if social rights are restricted, or not. So, we have: Safe(c,A), Safe(c,$\neg$A), Safe(c,C) and Safe(c,$\neg$C).

Let's consider now a situation where the database content is the result of the following insertions: a has inserted $A \rightarrow B$ and $B \rightarrow D$, b has inserted $B \wedge C \rightarrow D$ and C, c has inserted A, and the information about agent reliability is represented by:

$\text{Safe}(a, A \rightarrow B)$, $\text{Safe}(a, A \rightarrow \neg B)$, $\text{Safe}(b, B \wedge C \rightarrow D)$, Safe(c,A), Safe(c,$\neg$A), Safe(c,C) and Safe(c,$\neg$C).

First it may be observed that fact C is not guaranteed to be true since it was inserted by agent b who is not known to be safe regarding C ; and $B \rightarrow D$ is not guaranteed to be true since it was inserted by the agent a, who is not known to be an expert in sociology.

Now, if we put to the database the query: "is it true that unemployment increases?", formally represented by: B?, the answer is "yes", since the database contains A and $A \rightarrow B$, and both of them are guaranteed to be true, then B is true. However if we put the query: "is it true that criminality increases?", formally represented by: D?, then we note that there are just two ways to derive D from the database. The first one is represented by the derivation: $A, A \rightarrow B, B, \mathbf{B} \rightarrow \mathbf{D}, D$ and the second one is: $A, A \rightarrow B, B, \mathbf{C}, B \wedge C \rightarrow D, D$. The first derivation contains $B \rightarrow D$ which is not guaranteed to be true, and the second one contains C which is also not guaranteed to be true. So, D is not guaranteed to be true, and must therefore be considered to be merely a belief of the database.

This little toy example exhibits some interesting features of reasoning about the safety of information. The first is that one and the same sentence may be assigned one or more epistemic statuses, which need to be made explicit in a logical formalization. For instance, the rule $A \rightarrow B$ may represent a database belief, or it may represent information guaranteed to be true, or it may represent a piece of information regarding which some agent is known to be reliable. A second important feature is that the system which manages the database has to keep trace of the agents who have performed the insertions, because information safety depends on the reliability of these agents. In particular - returning to our example - even though the rule $B \wedge C \rightarrow D$ is logically redundant with respect to $B \rightarrow D$ (assuming that "$\rightarrow$" validates strengthening of the antecedent), we should not remove it, since it is guaranteed to be true, whilst B → D is merely a database belief.

We have developped for this purpose a logical framework in [4] which is recalled in the next section. Some possible options have been selected, whose results lead to the definition of the S logic where insertions are represented by a particular action operator. In section 2.2 a simplified version of S, the S' logic, ignores insertion actions, and only represents the effect of these actions, that is the inserted sentences and the agents who have performed the insertions. A second simplification, presented in section 2.3, is to only consider two modalities: EB for the representation of database explicit beliefs and B for database implicit beliefs. That leads to a third logic S". The axiomatics of the logic S, S' and S" are presented. We present the semantics of S", and it is proved that S" is valid and complete.

Then we present in section 3 an operational view of S" in terms of First Order Logic, where meta predicates are intended to represent the same information as modalities EB and B. A corresponding Prolog meta program is given. It is also proved that answers derived in S" and computed by the meta program are identical.

The main contribution of the paper is to show the different steps from a general logical formalization of information safety to an implemented Prolog program whose results have a clear meaning defined in a modal logical framework. Moreover this program is a resulting tradeoff between the generality of a powerful logical framework that was defined to help to understand complex phenomena, and the efficiency of an implementation that can manage non trivial applications.

2 Axiomatic definition of the logic

The concept of Safety is deeply related to the reliability of the sources of information. For this reason we have adopted the general framework of signalling acts [7], in order to have an explicit representation of the different types of agents involved in the process of information storage and retrieval. In our approach the DB is considered as a means for communication. Some agents, called "information sources", which may be users or sensors, bring it about that messages are stored in DB. These messages can be read by another agent, namely the DB management system, called the "system" for short, who knows the meaning of every stored message. The system is able to derive consequences of the information read in DB in order to compute answers to standard queries. There is another agent, called DB administrator, who plays a special role in the communication process. He is the agent who has information about source reliability. This meta-information, called Safety information, is stored by the administrator in a meta-database MDB, and it can be read by the system to compute answers to Safety queries.

A general formalization is presented that can also be used to investigate other issues than computing answers to Safety queries, such as database updates, or other issues related to database security.

Action operator for signalling acts.

An action modality E is introduced to represent acts of transmitting messages to the DB. Wffs of the form $E_a p$ will be read "the agent a brings it about that p", where p is a sentence about messages that are stored in DB. In the case where p is an atomic formula, it is of the form : in.DB(m), and its intended meaning is "the message m is stored in DB". E is closed under logical equivalence:

$$(RE)\ \frac{p_1 \leftrightarrow p_2}{E_a p_1 \leftrightarrow E_a p_2}$$

and it will be a "success" operator, in the sense that all instances of the schema (ET) are assumed true :

$$(ET) \quad E_a p \rightarrow p$$

Since we take tautologies to be outside the scope of anyone's agency, we accept the schema (E¬N) [9]:

$$(E\neg N) \quad \neg E_a \top \qquad (\text{where } \top \text{ is any tautology})$$

Since we accept (RE) and (R¬N) we must reject the distibution principle: (Em) $E_a(p_1 \wedge p_2) \rightarrow (E_a p_1 \wedge E_a p_2)$. A case can be made (cf. Elgesem, [5] for discussion of this), for accepting the converse principle:
(Ec) $(E_a p_1 \wedge E_a p_2) \rightarrow E_a(p_1 \wedge p_2)$.

Interpretation of signalling acts by the system.

We must find a way of representing how the result of each signalling act (the messages stored in DB) is interpreted by the system.

In a general approach we can consider that messages have no structure, and that they might be just strings of characters, or strings of bits. In that case the meaning of each message m_i is represented by an axiom (S_i) of the form :

$$(S_i) \quad K_s(E_a p_i \rightarrow B_a q_i)$$

If p_i is the sentence : in.DB(m_i), axiom (S_i) can be rephrased as "the system knows that, if the agent a brings it about that the message m_i is stored in DB, then a believes q_i". Formulas of the form $E_a p_i \rightarrow B_a q_i$ specify, in their consequents, the meanings assigned to signalling acts by the agent a.

In most of the existing databases the autonaming convention is adopted. That is, sentences are represented by strings of characters that are the sentences themselves. In this situation, if 'q' is a message the system interprets as meaning that q, then the meaning of messages may be represented by the unique axiom schema (S) instead of a set of axioms (S_i) :

$$(S) \quad K_s(E_a(\text{in.DB}('q')) \rightarrow B_a q)$$

For simplicity the core modality K will be defined by : $K_a q \overset{\text{def}}{=} (B_a q) \wedge q$, and the modality B will be assigned a logic of type KD45 [2].

We have assumed that the system is observing the DB, and that it is informed about every signalling act performed by information-transmitting sources. In other words, the system knows the DB content, and who stored the messages in DB. This assumption is formally represented by the axiom schema :

$$\text{(OBS)} \quad E_a p \rightarrow K_s E_a p$$

It may also be necessary for the system to collate the beliefs of the various agents who have inserted messages into DB, and to draw conclusions from this collection of beliefs. Where a is any agent, we thus introduce :

$$\text{(BEL)} \quad K_s B_a q \rightarrow K_s B q$$

where "$K_s Bq$" is read "the system knows that it is believed that q".

Concept of Safety.

Now the concept of Safety is defined by :

$$\mathrm{Safe}(a, p, q) \overset{\mathrm{def}}{=} K_{adm}(E_a p \rightarrow q)$$

where "q" is the meaning of the signalling act "$E_a p$".

Sentences of the form Safe(a,p,q) can be read "the agent a is reliable when he performs an action whose result is p, and whose meaning is q". The formal definition of Safe(a,p,q) means that the administrator, called "adm", knows that if the agent a brings it about that p then q is true.

If the autonaming convention is accepted the definition of Safe takes the form:

$$\mathrm{Safe}(a, q) \overset{\mathrm{def}}{=} K_{adm}(E_a(\mathrm{in.DB}('q')) \rightarrow q)$$

The axiom schema (SAF) says that all the safety information is known by the system, or, in other words, that the system knows the content of the metadatabase MDB :

$$\text{(SAF)} \quad K_{adm}(E_a p \rightarrow q) \rightarrow K_s(E_a p \rightarrow q)$$

2.1 Axiomatics of the S logic

The logic we have adopted will be called the S logic. In summary, it is formally defined by the following axiom schemas and inference rules:

- All the axiom schemas of Classical Propositional Calculus.
- For any modality of the form B_a, and for B, we have the axiom schemas of KD45, and any modality of the form K_a is defined as $K_a q \overset{\mathrm{def}}{=} (B_a q) \wedge q$.
- For the modality E we have:
  - (RE) $\frac{p_1 \leftrightarrow p_2}{E_a p_1 \leftrightarrow E_a p_2}$
  - (ET) $E_a p \rightarrow p$
  - (E¬N) $\neg E_a \top$

- The links between the modalities E_a, B_a, K_s and K_{adm} are defined by:
  - (OBS) $E_a p \rightarrow K_s E_a p$
  - (S) $K_s(E_a(in.DB('q')) \rightarrow B_a q)$
  - (BEL) $K_s B_a q \rightarrow K_s Bq$
  - (SAF) $K_{adm}(E_a p \rightarrow q) \rightarrow K_s(E_a p \rightarrow q)$
- The inference rules Modus Ponens and (for all modalities except E_a) Necessitation.

The semantics of the S logic is not presented here, but the semantics of each modal operator is well defined in terms of Kripke models or minimal modal models. For more details see [2].

Assumptions about agent safety are represented by sentences of the form Safe(a,q), which are defined by:

$$\mathrm{Safe}(a, q) \overset{\mathrm{def}}{=} K_{adm}(E_a(in.DB('q')) \rightarrow q)$$

A given state of the database is represented by the conjunction of a set of assumptions st defining safety of agents, and the insertion actions that have been performed. Then st is the conjunction of a set of sentences of the form:

$st = \{ \mathrm{Safe}(a, q_1),\ \mathrm{Safe}(a, q_2),\ \ldots, \mathrm{Safe}(b, q'_1),\ \mathrm{Safe}(b, q'_2), \ldots$
$E_a(In.DB('q'_i)),\ E_a(In.DB('q'_j)), \ldots, E_b(In.DB('q'_k)),\ E_b(In.DB('q'_l)),\ \ldots \}$

For a query represented by the sentence q, the answer to the standard query is "yes" iff we have: $\vdash_S st \rightarrow K_s Bq$, and the answer to the safety query is "yes" iff we have: $\vdash_S st \rightarrow K_s q$.

2.2 Axiomatics of the S' logic

We shall now define a simplified version of the S logic, called S', which will be an intermediate step toward the logic S" ; the latter will be powerful enough to represent the phenomena we want to consider in the context of Data and Knowledge Bases in this paper.

The first simplification in the definition of the S' logic is to consider only the result of insertion actions performed by the agents. So, the action operators of the form E_a are omitted, and sentences of the form: $E_a(In.DB('q'))$ are replaced by sentences of the form $EB_a q$, where EB_a is a new modality. $EB_a q$ can be read "the database explicitly believes q, and a sentence whose meaning is q has been inserted by the agent a". A consequence of this interpretation is that formulas in the scope of the EB_a modality are restricted to propositional formulas. To reflect the fact that in $EB_a q$ we do not consider the syntactical form of q, but only its meaning, we have to accept the following inference rule:

$$(\mathrm{RE'})\ \frac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$$

However the modality $EB_a q$ obeys none of the axiom schemas of KD45. The consequence of this first simplification is that (RE), (ET), (E¬N), (OBS), (S) and (SAF) must be removed, since each involves the operator E_a.

The second simplification is to drop the distinction between the agents called "administrator" and "system", replacing the modalities K_s and K_{adm} by the one modality K_s.

To summarize the formal definition of the S' logic is:

- All the axiom schemas of Classical Propositional Calculus.
- For the modalities B and B_s we have the axiom schemas of KD45, and the modality K_s is defined as $K_s q \overset{def}{=} (B_s q) \wedge q$.
- For each modality of the form EB_a we have:
  (RE') $\frac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$
- The link between the modalities EB_a, B and K_s is defined by:
  (BEL') $K_s(EB_a q) \rightarrow K_s(Bq)$
- The inference rules Modus Ponens and (for all modalities except EB_a) Necessitation.

Assumptions about the reliability of agents are now represented by sentences of the form Safe'(a,q) which are defined by:

$$\text{Safe}'(a,q) \overset{def}{=} K_s(EB_a q \rightarrow q)$$

A given state of the database is now defined by the conjunction of a set st' of sentences of the form:

st' = { $\text{Safe}'(a,q_1)$, $\text{Safe}'(a,q_2)$, $\ldots$, $\text{Safe}'(b,q_1')$, $\text{Safe}'(b,q_2')$, $\ldots$ $K_s(EB_a q_i)$, $K_s(EB_a q_j)$, $\ldots$, $K_s(EB_b q_k)$, $K_s(EB_b q_l)$, $\ldots$ }

For a query represented by the sentence q the standard answer is "yes" iff we have: $\vdash_{S'} st' \rightarrow K_s Bq$, and the answer to the safety query is "yes" iff we have: $\vdash_{S'} st' \rightarrow K_s q$.

2.3 Axiomatics of the S" logic

It is noticeable that in the S' logic derivation of formulas of the form $K_s Bq$ and $K_s q$ from st' only involves formulas that are prefixed by the modality K_s. That means that these derivations reflect derivations performed by the agent called the "system". So, the fact that we are in the context of the "system" knowledge could be understood to be implicit in these derivations, and that leads to a further simplification where the modality K_s is abandoned. Then the formal definition we get for the S" logic is:

- All the axiom schemas of Classical Propositional Calculus.
- For the modality B we have the axiom schemas of KD45.
- For each modality of the form EB_a we have:
  (RE') $\frac{q_1 \leftrightarrow q_2}{EB_a q_1 \leftrightarrow EB_a q_2}$
- The link between the modalities EB_a and B is defined by:
  (BEL") $EB_a q \rightarrow Bq$
- The inference rules Modus Ponens and Necessitation for the modality B.

Assumptions about the reliability of agent are represented in the S" logic by sentences of the form Safe"(a,q) which are defined by:

$$\mathrm{Safe}''(a,q) \overset{\mathrm{def}}{=} EB_a q \rightarrow q$$

It might be noticed that from $EB_a q$ and (BEL") we can infer Bq, and from the axiom schema (D) we have $\neg B(\neg q)$, and from the contrapositive form of (BEL") for $\neg q$, we have $\neg EB_a(\neg q)$. Then, due to properties of material implication we have Safe"(a,$\neg$q). It is quite counter-intuitive to be able to infer that agent a is reliable in regard to $\neg$q from the fact that a has inserted q in the database, and this shows that the definition of Safe" is not a correct formal representation of the concept of safety.

However, from an operational point of view, this fact has no dramatic consequences, in as much as we are not interested in deriving from st" conclusions regarding agent reliability. Indeed, if we only consider the derivation of answers to queries, to infer $\neg$q from Safe"(a,$\neg$q), we have to have $EB_a(\neg q)$, which together with $EB_a q$ leads to an inconsistency.

We do not have this problem with the definition of Safe' because to have Safe'(a,$\neg$q), $\neg EB_a(\neg q)$ has to be true in every world where the knowledge of the system S' is true, and not just in one particular world.

A given state of the database is defined by the conjunction of a set st" of sentences of the form:

st" = $\{ \mathrm{Safe}''(a,q_1), \mathrm{Safe}''(a,q_2), \ldots, \mathrm{Safe}''(b,q'_1), \mathrm{Safe}''(b,q'_2), \ldots$
$EB_a q_i, EB_a q_j, \ldots, EB_b q_k, EB_b q_l, \ldots \}$

For a query represented by the sentence q the standard answer is "yes" iff we have: $\vdash_{S''} st'' \rightarrow Bq$, and the answer to the safety query is "yes" iff we have: $\vdash_{S''} st'' \rightarrow q$.

2.4 Semantics of the S" logic

A model M for the S" logic is a tuple $M =< W, R, f_{a_1} f_{a_2}, \ldots, f_{a_n}, P >$, where :

- W is a non empty set of worlds,
- R is a relation on WxW, transitive and euclidean,
- each f_{a_i} is a function from W to 2^{2^W}, and
- P is a function from propositional variables to 2^W.

The following constraint C_i is imposed to each function f_{a_i}:
C_i : if r(w) denotes the set of worlds {w' : wRw' }, then for each X such that $X \in f_{a_i}(w)$ we must have $r(w) \subseteq X$.

This constraint is intended to validate the (BEL") axiom schema.

The satisfiability relation is defined as usual:

$M, w \models p$	iff	$w \in P(p)$, if p is a propositional variable
$M, w \models \neg p$	iff	$M, w \not\models p$
$M, w \models p \vee q$	iff	$M, w \models p$ or $M, w \models q$
$M, w \models Bp$	iff	$\forall w'(wRw' \Rightarrow M, w' \models p)$
$M, w \models EB_{a_i} p$	iff	$\|p\| \in f_{a_i}(w)$

where $\|p\|$ denotes the truth set of p: $\{ w' : M, w' \models p \}$.

The notion of valid formulas is defined as usual by :

$\models_{S''} p$ iff $\forall M = < W, R, .., P >, \forall w \in W, M, w \models p$

Theorem 1. *The S" logic is valid and complete.*

Proof. The proof of validity is obvious. The proof of completeness uses the technique of canonical minimal models [2].

2.5 Links between S, S' and S"

To show how the S' logic relates to the S logic we define a transformation T that assigns to any sentence of the form Safe(a,q) the sentence Safe'(a,q), and to any sentence of the form $E_a(In.DB('q'))$ the sentence EB_aq. If T is extended to sets of sentences in the natural way, then, if st'=T(st) we should have the property:

$$\vdash_S st \rightarrow K_sBq \quad \text{iff} \quad \vdash_{S'} st' \rightarrow K_sBq, \quad \text{and :} \quad \vdash_S st \rightarrow K_sq \quad \text{iff} \quad \vdash_{S'} st' \rightarrow K_sq$$

To show how the S" logic relates to the S' logic we define a transformation T' that assigns to sentences of the form Safe'(a,q) sentences of the form Safe"(a,q), and to sentences of the form $K_s(EB_aq)$ sentences of the form EB_aq. Then, if st"=T'(st') we should have the property:

$$\vdash_{S'} st' \rightarrow K_sBq \quad \text{iff} \quad \vdash_{S''} st'' \rightarrow Bq, \quad \text{and :} \quad \vdash_{S'} st' \rightarrow K_sq \quad \text{iff} \quad \vdash_{S''} st'' \rightarrow q$$

Notice that the above two properties require only that if we are in the same context we should get the same answers whether we are in the S logic, or in the S' logic, or in the S" logic.

In our work to date we have not formally proved these two properties. So, we can only consider the S logic and the S' logic as intuitive justifications for the definition of the S" logic, and accept the S" logic as a basis for the definition of answers to standard or safety queries.

3 Operational definition

In this section is specifyied, at the meta level, a theorem prover that allows us to generate answers to standard queries as well as answers to safety queries addressed to a database. This specification is defined in First Order Logic by Horn clauses and their translation in Prolog is a simple exercise. An important feature is that the modalities EB and B of S" are respectively represented by two meta predicates Bexp and B, and true beliefs of S" are represented by the meta predicate K.

We assume that the agents do not insert formulas in the database, but they insert sets of clauses which represent these formulas. In the same way, we assume that the reliability of agents is not described in terms of propositional formulas but in term of sets of clauses.

We note cl(f) the set of clauses which represent a formula f. We assume that the function cl satisfies the following property: two equivalent formulas are associated by cl with two sets of clauses such that any clause in one set is variant of a clause in the other set (.i.e has the same literals than that clause).

We do not restrict ourselves to Horn clauses but we allow agents to insert sets of general clauses.

The prover described in this section, is based on the prover which has been defined previously in [1], for general clauses : the main trick is to transform each general clause into a set of Horn clauses, by renaming negative literals, and to take the factorisation into account.

The specification is given by means of meta axioms presented in 3.1. In 3.2, we show how to represent the database and the queries at the meta level. The soundness and completeness of the meta axioms are presented in 3.3.

3.1 The meta axioms

Let us consider the following meta axioms where all the variables are assumed to be universally quantified:

(1) $\mathrm{in}(\mathrm{l}, \mathrm{s}) \rightarrow \mathrm{B}(\mathrm{s}, \mathrm{l})$

(2) $\mathrm{Bexp}(\mathrm{a}, \mathrm{sc}) \wedge \mathrm{in}(\mathrm{conj} \rightarrow \mathrm{q}, \mathrm{sc}) \wedge \mathrm{comp}(\mathrm{q}, \mathrm{q}') \wedge \mathrm{Bconj}(\mathrm{q}' \wedge \mathrm{s}, \mathrm{conj}) \rightarrow \mathrm{B}(\mathrm{s}, \mathrm{q})$

(3) Bconj(s,true)

(4) $\mathrm{B}(\mathrm{s}, \mathrm{b}_1) \wedge \mathrm{Bconj}(\mathrm{s}, \mathrm{b}) \rightarrow \mathrm{Bconj}(\mathrm{s}, \mathrm{b}_1 \wedge \mathrm{b})$

(5) $\mathrm{in}(\mathrm{l}, \mathrm{s}) \rightarrow \mathrm{K}(\mathrm{s}, \mathrm{l})$

(6) $\mathrm{Bexp}(\mathrm{a}, \mathrm{sc}) \wedge \mathrm{Safe}(\mathrm{a}, \mathrm{sc}) \wedge \mathrm{in}(\mathrm{conj} \rightarrow \mathrm{q}, \mathrm{sc}) \wedge \mathrm{comp}(\mathrm{q}, \mathrm{q}') \wedge \mathrm{Kconj}(\mathrm{q}' \wedge \mathrm{s}, \mathrm{conj}) \rightarrow \mathrm{K}(\mathrm{s}, \mathrm{q})$

(7) Kconj(s,true)

(8) $\mathrm{K}(\mathrm{s}, \mathrm{b}_1) \wedge \mathrm{Kconj}(\mathrm{s}, \mathrm{b}) \rightarrow \mathrm{Kconj}(\mathrm{s}, \mathrm{b}_1 \wedge \mathrm{b})$

The predicate B(s,q) (resp. K(s,q)) means that s → q is a database belief (resp. a true database belief). Comp(q,q') means that q' is a complementary literal of q. The meta axiom (1) says that if l is a literal in the clause s, then s → l is a database belief. The meta axiom (2) says that if conj → q is in the set of agent'a explicit beliefs and $\neg \mathrm{q} \wedge \mathrm{s} \rightarrow \mathrm{conj}$ is a database belief, then s → q is a database belief. Axiom (3) says that s → true is a database belief. Axiom (4) says that if s → b and $\mathrm{s} \rightarrow \mathrm{b}_1$ are database beliefs then $\mathrm{s} \rightarrow \mathrm{b}_1 \wedge \mathrm{b}$ is a database belief. Axioms (5) to (8) have a similar meaning as axioms (1) to (4).

3.2 Description of the database and of the query at the meta level

Horn clauses associated to a sentence. A sentence q is represented by its clausal form cl(q), and clauses are represented by sets of Horn clauses by renaming negative literals. This set of Hornclauses is denoted by H(cl(q)). For instance, if q is the sentence $\mathrm{A} \vee \mathrm{B}$, we have: $\mathrm{H}(\mathrm{cl}(\mathrm{q}))=\{\mathrm{notA} \rightarrow \mathrm{B}, \mathrm{notB} \rightarrow \mathrm{A}\}$.

State of affairs and query. Let us consider a propositional formula f, which represents a query asked by a user. We introduce in L a new propositional variable, noted Q. Let us recall that cl(f↔Q) denotes the clausal form of the formula f↔Q.

If st" is a set of sentence of the form: st" $= \{ \mathrm{Safe}''(a, q_1), \mathrm{Safe}''(a, q_2), \ldots, \mathrm{Safe}''(b, q'_1), \mathrm{Safe}''(b, q'_2), \ldots \mathrm{EB}_a q_i, \mathrm{EB}_a q_j, \ldots, \mathrm{EB}_b q_k, \mathrm{EB}_b q_l, \ldots \}$ where q_is and q'_is are propositional formulas, we define the set meta(st",f) from st" and f by replacing Safe(a,q) in st" by the sentence Safe(a, H(cl(q)) and by replacing EB_a(q) by Bexp(a, H(cl(q)), and by inserting Safe(user, H(cl(f ↔ Q))) and Bexp(user, H(cl(f ↔ Q))). So, in formal terms we have:

$$\begin{aligned} \mathrm{meta}(st'',f) = & \{ \mathrm{Safe}(a, H(cl(q))) : \mathrm{Safe}(a,q) \in st'' \} \cup \\ & \{ \mathrm{Bexp}(a, H(cl(q))) : \mathrm{EB}_a q \in st'' \} \cup \\ & \{ \mathrm{Safe}(user, H(cl(f \leftrightarrow Q))), \ \mathrm{Bexp}(user, H(cl(f \leftrightarrow Q))) \} \end{aligned}$$

In other terms, meta(st",f) contains the meta information which represent :

- the fact that such an agent has inserted such a formula,
- the fact that such an agent is safe as regard to such a formula,
- the fact that the user is asking such a query and
- the fact that this user is considered to be safe as regard to his formulation of the query.

3.3 Soundness and completeness of the meta axioms

Theorem 2. *Let f be a formula L. Let Q be the new propositional variable introduced previously, we have:*

$\vdash_{S''} st'' \rightarrow Bf \;\Leftrightarrow\; \{(1), \ldots, (8)\} \vdash \mathrm{meta}(st'', f) \rightarrow B(true, Q)$, *and*

$\vdash_{S''} st'' \rightarrow f \;\Leftrightarrow\; \{(1), \ldots, (8)\} \vdash \mathrm{meta}(st'', f) \rightarrow K(true, Q)$

Proof. We have shown that deriving, in S" logic, formulas like Bf or f from st" comes to derive, in classical logic, formula f from the set of clauses inserted by the agents, and we re-use properties of a prover that we had defined in previous work [1].

4 Conclusion

We have presented a general logical framework for reasoning about the safety of information stored in a database, and its simplified version, the S" logic, to derive answers to standard queries and safety queries. The S" logic allows to derive different safety answers from several databases that represent the same theory in different syntactical forms. For instance answers derived from a database that contains $\mathrm{EB}_a(p \wedge q)$ are not necessarily the same as answers derived from another database containing $EB_a p$ and $EB_a q$. However $EB_a(p \wedge q)$ and $EB_a(q \wedge p)$ lead to the same answer. That means that the insertion of two sentences that represent the same proposition are considered to be equivalent.

The operational view of the S" logic is relatively simple and efficient since it is defined by Horn clauses in First Order Logic. There is a limited restriction about the form of inserted sentences since they must be a conjunction of clauses. The reason for this is to avoid to check equivalence of two sentences p and p' that occur in $EB_a p$ and in Safe(a,p'), in the general case.

In future works it will be worth investigating potential applications of the S" logic to the problem of belief revision. Indeed database representation is not purely syntactic like in [8] and it is not a belief set like in [6].

Acknowledgement : This work was partially supported by the ESPRIT BRA project MEDLAR2.

References

1. A. Bauval and L. Cholvy. Automated reasoning in case of inconsistency. In *Proceedings of WOCFAI*, Paris, France, 1991.
2. B. F. Chellas. *Modal Logic: An introduction.* Cambridge University Press, 1988.
3. R. Demolombe and A. Jones. Integrity Constraints Revisited. In A. Olive, editor, *4th International Workshop on the Deductive Approach to Information Systems and Databases.* Universitat Politecnica de Barcelona, 1993.
4. R. Demolombe and A. Jones. Deriving answers to safety queries. In R. Demolombe and T. Imielinski, editor, *Non Standard Queries and Answers*, Oxford, To appear. Oxford University Press.
5. D. Elgesem. *Action Theory and Modal Logic.* PhD thesis, University of Oslo, Department of Philosophy, 1992.
6. P. Gardenfors. *Knowledge in flux : modeling the dynamics of epistemic states.* The MIT Press, 1988.
7. A. Jones. Toward a Formal Theory of Communication and Speech Acts. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communications.* The MIT Press, 1990.
8. G.M. Kupper, J.D. Ullman, and M. Vardi. On the equivalence of logical databases. In *Proc of ACM-PODS*, 1984.
9. I. Porn. Action Theory and Social Science. Some Formal Models. *Synthese Library*, 120, 1977.
10. R. Reiter. What Should a Database Know? *Journal of Logic Programming*, To appear.

Classical Methods in Nonmonotonic Reasoning*

Yannis Dimopoulos

Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
yannis@mpi-sb.mpg.de

Abstract. In this paper we present and compare some classical problem solving methods for computing the stable models of a general propositional logic program. In particular linear programming, propositional satisfiability, constraint satisfaction, and graph algorithms are considered. Central to our approach is the representation of a logic program by means of a graph.

1 Introduction

Over the last years, a large body of research has been devoted to the semantics of logic programs with negation. One of the most prominent proposals is the *stable model* semantics, introduced in [10]. Unfortunately, stable models semantics turned out to be intractable, even in simple cases. Despite their computational hardness, a relatively small effort has been expended on deriving effective ways for reasoning with these formalisms. Recently, some classical problem solving methods for computing the semantics of logic programs and default theories have been proposed ([3], [4], [1], [2], [8], [9]). These include linear programming, propositional satisfiability, constraint satisfaction, and graph algorithms.

In this paper we examine and compare these different methods. Central to our approach is the representation of a general logic program by means of a graph. The idea was introduced in [8] for the case of disjunction free default theories. The set of stable models of a logic program corresponds to a subset of the *kernels* of the associated graph. The graph model gives rise to four different methods for computing the stable models of a logic program.

The first alternative is algorithms that explicitly enumerate the kernels of the graph. The second possibility is to express the graph structure in terms of propositional logic, and use a satisfiability algorithm. The models of the resulting theory coincide with the set of models of the Clark's completion of the program. As an alternative, a variant of the Davis-Putman method that computes exactly the stable models, is presented.

The third method is the formulation of the problem as a Linear Programming one, while the last is the use of some of the constraint satisfaction algorithms.

Finally, we present our computational experience with some implementations of the satisfiability, linear programming and graph methods.

* This paper is an extended abstract of [7].

2 Preliminaries

A (general) propositional logic program is a finite set of rules of the form $A \leftarrow B_1 \wedge \ldots \wedge B_n \wedge \neg C_1 \wedge \ldots \neg C_m$ where $A, B_1, \ldots B_n, C_1, \ldots C_m$ are atoms. The lhs of the rule (i.e. the atom A) is called the head while the rhs is called the body. We call a logic program *negative* if the bodies of all the rules contain only negative literals.
Let P be a logic program and M an interpretation. We define $G_M(P)$ to be the logic program obtained by first deleting every rule with a negative literal that occurs in its body and does not belong to M, and then deleting all negative literals from the remaining rules. See that the resulting program $G_M(P)$ has a unique minimal model.

Definition 1. ([10]) Let P be a logic program and M an interpretation. Then M is a *stable model* for P if the minimal model of $G_M(P)$ coincides with M. □

One of the early attempts to define the semantics of a logic program with negation is Clark's predicate completion ([6]). The completion of a propositional logic program P, denoted by *comp*(P), is obtained in two steps

- Replace every rule of the form $A \leftarrow B_1 \wedge \ldots \wedge B_n \wedge \neg C_1 \wedge \ldots \neg C_m$ by the implication $A \leftarrow B_1 \wedge \ldots \wedge B_n \wedge \sim C_1 \wedge \ldots \sim C_m$, where $\sim$ denotes classical negation.
- Let $Q \leftarrow Body_1, \ldots, Q \leftarrow Body_k$ be all the clauses with Q in the head. If the clause $'Q \leftarrow'$ belongs to this set then replace this set by Q. Otherwise replace the set of clauses by $Q \leftrightarrow Body_1 \vee \ldots \vee Body_k$. If Q occurs nowhere in the head of the implications add $\sim Q$ in *comp*(P).

Let $G = (N, E)$ be a directed graph where N is the set nodes and E the set of edges. Then for $n_i \in N$, we define $\Gamma^+(n_i) = \{n_j | (n_i, n_j) \in E\}$ and $\Gamma^-(n_i) = \{n_j | (n_j, n_i) \in E\}$. The basic graph theoretic concept used in this paper is that of the kernel of a graph.

Definition 2. Let $G = (N, E)$ be a directed graph. A set of nodes $K \subseteq N$ is a *kernel* for G if for every two nodes $n_i, n_j \in K$, the edges (n_i, n_j) and (n_j, n_i) do not belong to E (such a set is called independent), and for every node $n_j \in N - K$ there is a node $n_i \in K$ such that $n_i \in \Gamma^-(n_j)$. □

3 Graphs for Logic Programs

If not otherwise stated, we assume that for every literal p or $\neg p$ which occurs in the body of a rule of a program P, the corresponding positive literal p occurs in the head of some rule in P. We call this class of programs *complete* logic programs. It is easy to see that every logic program can be transformed to a complete one in polynomial time.

We now briefly introduce the way the *rule graph* $G_P = (N, E)$, of a logic program P is constructed (for a detailed discussion see [8]). The set of nodes is $N = R \cup A$, $R = \{r_i | r_i$ is a rule of $P\}$ and $A = \{a_i|$ for each atom a_i of $P\}$. The set $E = \{(r_i, r_j)| \neg p \in \text{body}(r_j)$ and $p \in \text{head}(r_i)\} \cup \{(a_i, r_j)|a_i \in \text{body}(r_j)\} \cup \{(r_i, a_j)|a_j = \text{head}(r_i)\}$.

We can prove that every stable model of P corresponds to a kernel of G_P. Namely, if M is a stable model for a program P, then there is a kernel K for the rule graph G_P such that for every $p \in M^+$ (M^+ denotes the set of atoms in a set of literals M) there is a node $r_i \in K$ such that $\text{head}(r_i) = p$. However the converse is not always true. This is because of the possible circular support between the rules. Consider for example the program $P = \{a \leftarrow b, b \leftarrow a\}$. Its rule graph is depicted in figure 1. The graph has two kernels $K_1 = \{r_1, r_2\}$ and $K_2 = \{a, b\}$. See that only the second kernel corresponds to a stable model of P, namely $M = \{\neg a, \neg b\}$.

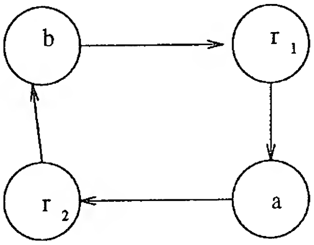


Figure 1

For a kernel K to correspond to a stable model, it must be the case that for the nodes of the set $K \cap R$, there exists a nonnegative integer function ϕ called *sequential valuation* such that $\phi_{r_i} = max_j(min_k(\phi_{r_{jk}}))$, where $r_{jk}, r_i \in K \cap R$, for every $a_j \in A$ and $(r_{jk}, a_j), (a_j, r_i) \in E$. Obviously if $\Gamma^-(r_i) \cap A = \emptyset$ then $\phi_{r_i} = 0$. We call the kernels complying with this property *sequential kernels.* Intuitively speaking, the value ϕ_{r_i} denotes the earliest step at which the rule r_i can be applied.

It turns out that the graph model is equivalent to Clark's completion, in the sense that the kernels of the rule graph are in direct correspondence with the models of the completed program, as stated by the following two theorems.

Theorem 3. *Let K be a kernel for G_P, and $S = \{p_i | \exists r_i, p_i \in head(r_i), r_i \in K \cap R\}$. Then the assignment $V(p_i) = true$ iff $p_i \in S$, is a model for comp(P).* □

Theorem 4. *Let V be a model for comp(P). Then the set $K = \{r_i | r_i \in R, \forall p_i \in body(r_i), V(p_i) = true\} \cup \{a_i | a_i \in A, V(a_i) = false\}$ is a kernel for G_P.* □

4 The Different Methods

We describe four different methods for computing the stable models of a program. In all the methods the input is a representation of the rule graph and the problem to be solved is to compute the kernels.

4.1 Satisfiability Algorithms

In this section we show how the problem of computing the stable models of a logic program can be expressed in terms of propositional logic. Let $G_P = (N, E)$ be the graph associated with a logic program P. The task is to compute the kernels of G_P. Our purpose is to construct a propositional theory T_P such that its atoms correspond to the nodes of G_P and its satisfying truth assignments to the kernels of G_P. The convention is that the set of nodes that are assigned the value true in a model of T_P must form a kernel for the associated graph and vice versa. If K is a kernel, then every node adjacent to some node $n_i \in K$ must not belong to the kernel, which in terms of propositional logic can be expressed by the implication $n_i \rightarrow \neg n_1 \wedge \ldots \wedge \neg n_k$ (or the set of clauses $\neg n_i \vee \neg n_1, \ldots, \neg n_i \vee \neg n_k$), for all $n_j \in \Gamma^+(n_i)$, $1 \leq j \leq k$. On the other hand if a node n_i does not belong to K then some node $n_j \in \Gamma^-(n_i)$ must be in K. This can again be expressed by the proposition $\neg n_i \rightarrow n_1 \vee \ldots \vee n_k$ for all $n_j \in \Gamma^-(n_i)$, $1 \leq j \leq k$. A clause of this form is said to be *indexed by* n_i. Then the theory T_P is defined to be the smallest set of clauses that subsumes the above set of implications. It is easy to prove that every model of T_P induces a kernel for the associated graph G, and vice-versa. Consequently, we can use a propositional satisfiability algorithm to compute the kernels of the graph. The size of T_P is given by the next proposition.

Proposition 5. *Let T_P be the propositional theory of a logic program P. The size of T_P, denoted as $\| T_P \|$, is less than $(r_P + 2) \times \| P \| + \| B_L \|$, where r_P is the maximum number of literals in the body of any rule in P, and B_L is the set of propositional atoms occurring in P.* □

However, the interesting satisfying truth assignments are only those that correspond to sequential kernels. In order to obtain this subset of models, we can employ two different methods. The first, and most straightforward method, is to compute all models of T_P by using a satisfiability algorithm, and then test each of them for stability in polynomial time.

The second possibility is to enhance a given propositional satisfiability algorithm with metarules that restrict the search to the interesting assignments. In the sequel of this section we present such a set of metarules for the Davis-Putman procedure and prove that the method is sound and complete wrt the set of sequential kernels and consequently the set of stable models.

The Davis-Putman algorithm (see [5]) is a sound procedure for propositional satisfiability consisting of the following 4 rules, which can be applied iteratively to simplify a set of clauses C:

1. *Tautology Rule*: Delete all clauses in C which are tautologies.
2. *One-literal Rule*: If there is a unit clause L in C, then assign the value true to L and delete all clauses in C containing L. If the resulting set of clauses, C', is empty then C is satisfiable. Otherwise delete from the clauses in C' all occurrences of $\neg L$.
3. *Pure-literal Rule*: If a literal L occurs in a clause of C and the literal $\neg L$ does not occur in any clause of C, then assign L the value true and delete all clauses containing L.

4. *Splitting Rule*: If the set C can be expressed in the form

$$(A_1 \vee L) \wedge (A_2 \vee L) \wedge \ldots \wedge (A_m \vee L) \wedge (B_1 \vee \neg L) \wedge (B_2 \vee \neg L) \wedge \ldots \wedge (B_n \vee \neg L) \wedge R$$

where R does not contain any of the L or $\neg L$ then split the search space into two, the first being $A_1 \wedge A_2 \wedge \ldots A_m \wedge R$ and the second $B_1 \wedge B_2 \wedge \ldots B_n \wedge R$. The first branch corresponds to the value assignment true to L, while the second to the assignment false.

If the above rules are applied iteratively, starting from a set of clauses C, and at each point where the splitting rule is used one of the two possible assignments to L is chosen, then if the empty set is derived the corresponding value assignment obtained is a model of C. If at some point a contradiction is reached the algorithm backtracks to an earlier splitting point and a different assignment is attempted. If all of the search paths fail then the set of clauses C is unsatisfiable. Since only some of the models of a given set of clauses T_P correspond to the stable models of P we need to restrict the search space by augmenting Davis-Putman method with metarules.

Given a logic program P we can easily obtain its positive counterpart P^+, by deleting all negative literals from the body of the rules. Then by constructing the rule graph of P^+, denoted as G_{P^+}, we can identify the strongly connected components of G_{P^+} and the associated directed acyclic graph (DAG) $G^S_{P^+}$. The graph $G^S_{P^+}$ induces an ordering on the components, such that $O(C_i) = max\{O(C_j) | C_j \in \Gamma^-(C_i)\} + 1$ (all nodes with in-degree 0 are assigned the value 1). We call this value *depth* of the component. Hence, every atom can be considered as parameterized by the depth of the component to which it belongs.

On the other hand Davis-Putman method is augmented with a priority among the four rules it consists of. The three first rules have equal priority which is higher than the priority of the splitting rule. This means that the splitting rule will be applied if none of the other three rules can be applied. Furthermore, the splitting rule is applied to a literal of the current top component C_i induced by the depth ordering. A particular literal from C_i is chosen as follows. If there exists a rule node $r_i \in R \cap C_i$, such that the clause indexed by r_i does not contain any literal node $a_i \in A$ then the splitting is performed upon r_i. If no such atom exists, then an atom $r_j \in R$, $r_j \in C_i$, is chosen and assigned the value false, while the branch with the assignment true to r_j is omitted.

At each point, the above method assigns to a subset of atoms occurring in the set of clauses at hand C, a truth value. This partial value assignment is called a *MDP-valuation* of C. A complete MDP-valuation for C which derives the empty set, and consequently is a model of C, is called a *MDP-model* for C.

Example 1. Consider the logic program P

$c \leftarrow \neg d$ (r_1) $\quad$ $d \leftarrow \neg c$ (r_2) $\quad$ $a \leftarrow d$ (r_3)
$b \leftarrow a$ (r_4) $\quad$ $a \leftarrow b$ (r_5)

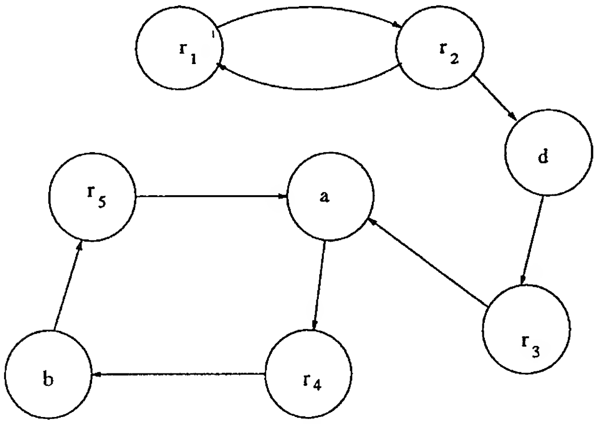


Figure 2

The rule graph of this program is depicted in Figure 2. The associated propositional theory is
$\neg r_1 \vee \neg r_2$, $\neg r_2 \vee \neg d$, $\neg d \vee \neg r_3$, $\neg r_3 \vee \neg a$, $\neg a \vee \neg r_4$,
$\neg r_4 \vee \neg b$, $\neg b \vee \neg r_5$, $\neg r_5 \vee \neg a$, $r_1 \vee r_2$, $d \vee r_2$,
$r_3 \vee d$, $a \vee r_3 \vee r_5$, $r_4 \vee a$, $b \vee r_4$, $r_5 \vee b$.
A possible ordering of the components of the initial graph is $(C_2, C_1, C_3, C_4, C_5)$, where $C_1 = \{r_1\}$, $C_2 = \{r_2\}$, $C_3 = \{d\}$, $C_4 = \{r_3\}$, $C_5 = \{a, r_4, b, r_5\}$. See that the assignment true to r_2 leads to the *MDP−model* $\{r_2, \neg r_1, \neg d, r_3, \neg a, r_4, \neg b, r_5\}$ while the assignment of the value false, leads to the *MDP−model* $\{\neg r_2, r_1, d, \neg r_3, \neg r_4, b, \neg r_5, a\}$. These models correspond to the stable models of P, $M_1 = \{a, b, \neg c, d\}$ and $M_2 = \{\neg a, \neg b, c, \neg d\}$ respectively. □

The following theorem demonstrates the correctness and completeness of the method wrt the sequential kernels, and consequently wrt the stable models.

Theorem 6. *Let P be a logic program, $G_P = (N, E)$ its rule graph, and T_P the associated propositional theory. Then a value assignment V to the literals is a MDP-model iff V induces a sequential kernel for G_P.* □

4.2 Linear Programming Algorithms

Since the kernel problem can be formulated in terms of propositional logic, the associated satisfiability problem can be solved by an integer linear programming algorithm. Any satisfiability problem can be express as a integer linear programming one where, to every literal A in the set of clauses a binary variable X_A is associated, the objective function is empty, while the set of constraints is the following:

1. For every clause $p_1 \vee p_2 \ldots \vee p_n \vee \neg k_1 \vee \neg k_2 \ldots \vee \neg k_m$, where p_i, k_j atoms, we add the constraint $\sum_{i=1}^{n}(1 - X_{p_i}) - \sum_{i=1}^{m} X_{k_i} \geq 1$.
2. For every variable X_A we add the constraints that it is a binary variable, that is, X_A can be only assigned the values 0 and 1.

Every solution for this set of constraints is a model for the associated set of clauses. Formulating T_P in linear programming takes $(r_P + 2)\times \parallel P \parallel + \parallel B_L \parallel$ constraints to represent each clause in T_P, plus $2\times \parallel P \parallel + 2\times \parallel B_L \parallel$ more to express the fact that these are 0-1 variables. This gives us a total of $(r_P + 4)\times \parallel P \parallel + 3\times \parallel B_L \parallel$, and leads to a simplex tableau of size $((r_P + 4)\times \parallel P \parallel + 3\times \parallel B_L \parallel) \times (\parallel P \parallel + \parallel B_L \parallel)$. We denote the set of constraints representing T_P as $ct(P)$. Again, not all of these assignments correspond to stable models. The following proposition determines a subset of models of T_P that includes the set of stable models of P.

Proposition 7. *Let K be a sequential kernel for the rule graph G_P of a program P. Then K is a minimal kernel wrt the set of rule nodes it contains.* □

Since the sequential kernels are in direct correspondence with the stable models, the models which are minimal wrt to the literals associated to rules, will include the stable models of the program P. Hence the objective function in this case becomes the minimization of the sum $\sum_{A\in R} X_A$. This is not exactly what we need, since only the assignments with the smallest number of variables from R will be computed, i.e. the assignments with the *minimal cardinality.* Our aim is to compute all the minimal assignments wrt set inclusion. Hence, similar to what is proposed in [2], we have to iterate this process adding at each step the constraint $\sum_{B\in M} X_B \leq (k-1)$, where M is the set of variables from R which have been assigned the value true during the last iteration, and k is the cardinality of M. This procedure will compute all the minimal models in order of non-decreasing cardinality. However, the converse of proposition 7 does not always hold, and the minimal models computed by this procedure must be tested for stability.

In [2] another linear programming formulation of the same problem is presented, based on the method for computing the minimal models of logic programs presented in [1]. The size of the simplex tableau of this formulation is, in the worst case, $((r_P + 3)\times \parallel P \parallel + 2\times \parallel B_L \parallel) \times ((r_P + 6)\times \parallel P \parallel + 3\times \parallel B_L \parallel)$, which is obviously larger than the size of the $ct(P)$ problem (see [7] for details).

4.3 Graph algorithms

Another alternative is to solve the kernel problem by a graph algorithm. A first attempt towards this direction is the algorithm presented in [9]. It is a backtracking algorithm with worst time complexity $O(p(N)2^{|F|})$, where F is a *feedback vertex set* of the input graph, and $p(N)$ a polynomial in the size of the input. A feedback vertex set is a set of nodes which when removed result in a acyclic graph. Since computing a feedback vertex set with minimum cardinality is a NP-hard problem approximation algorithms can be used leading to a, hopefully small, feedback vertex set. According to this method, given a program P the rule graph G_P is constructed and the graph algorithm computes the kernels of G_P, which are tested in order to determine whether they are sequential.

Despite the hardness of the general case, a class of logic programs is easier to reason with. Namely, in [9] an algorithm for computing a subset of the kernels of a odd cycle free graph is presented. The basic graph-theoretic property used is that every strongly connected graph with no odd cycles is bipartite. A graph $G = (N, E)$ is called *bipartite* if its nodes can be split into two parts, say K, L such that $E \subseteq (K \times L) \cup (L \times K)$. See that any of the two sets K, L is a kernel for G. Furthermore, this subset of the kernels can be computed with *polynomial delay*, that is, there is a polynomial (in the size of the input) delay between computing two consecutive kernels (see [9] for details).

Using this graph property we obtain a procedure which computes a sequential kernel of an odd loop free graph G_P associated with a logic program P. Following [11], we call a set of literal nodes U an *unfounded set* if the subgraph of G_{P+} induced by U and their preceding rule nodes has no source.

```
**Compute-Sequential-Kernels**($G_P, K$);
if $G_P = \emptyset$ then return($K$) else
  begin
  Compute the top components of the graph $G_P$, say $C_1, \ldots, C_k$;
  For i:=1 to k do;
      begin
        if $C_i$ contains an unfounded set $U_i$ then
        Add every node $n_i \in U_i \cap L$ to $B$
        else
        begin
        if $|C_i| = 1$ then add $C_i$ to $B$ else
        begin
        Split $C_i$ into two components $C_{i1}$ and $C_{i2}$;
        Add each node of $C_{i1}$ to $B$;
        end;
        end;
      end;
  $K' = K \cup B$;
  $N' := N - (B \cup \{n_i | n_i \in \Gamma^+(B)\})$;
  $G'_P := (N', E')$;
  Call Compute-Sequential-Kernels($G'_P, K'$);
  end;
```

Theorem 8. *Let P be a program and G_P its odd cycle free rule graph. Then the above algorithm computes a sequential kernel of G_P, in polynomial time.* □

Moreover, if the above algorithm is modified to consider both partitions of the components at each point, then all the "standard" sequential kernels can be computed with polynomial delay.

Corollary 9. *A subset of the stable models of a program with no odd cycles can be computed with polynomial delay.*

4.4 Constraint Satisfaction Algorithms

In [4], a translation of a restricted class of disjunctive logic programs to propositional logic is presented. This class includes the general logic programs considered here, and the translation is similar to this proposed by the authors for default logic ([3]). Except from satisfiability algorithms, they propose the use of constraint satisfaction graph-based methods as *tree-clustering* and *cycle-cutset decomposition*, for solving the associated satisfiability problems. Here we briefly outline the cycle-cutset decomposition, because of its similarity to the graph algorithm mentioned in the previous section.

The cycle-cutset decomposition method consists of first computing a propositional theory D_P whose models are in direct correspondence with the stable models of P and then building the undirected *constraint graph* of D_P. The constraint graph is identical to another *undirected* graph, namely the *interaction graph*, I_P, which is constructed directly from the program at hand P, as follows. Every atom of P is associated with a node of I_P, while for every atom p, the set of all atoms that occur in rules that have p in their heads, are connected as a clique (complete subgraph).

The decomposition method then identifies a cycle cutset for I_P, which is just another name for the feedback vertex set, assigns values to the literals in the feedback vertex set, and then propagates those values to the rest of the graph in polynomial time. This leads to an algorithm which is exponential in the size of the feedback vertex set. The idea is essentially the same with this of the graph algorithm of the previous section. On the other hand, the two graphs on which the methods operate, namely G_P and I_P, are not the same, hence their cycles and therefore their worst time complexity may differ. We call a program P *rule circular* if all of the rules residing in a cycle of G_P have different heads. For this class of programs the following proposition holds.

Proposition 10. *Let P be a rule circular logic program and G_P and I_P its rule and interaction graphs respectively. Then for every cycle in G_P that contains more than two rule nodes, there is a corresponding cycle in I_P.* □

Note that the last proposition does not necessarily mean that for every different cycle in G_P there is a different cycle in I_P, since more than one cycles of G_P can be mapped to a single cycle of I_P. However, if in P each atom occurs in the head of at most one rule, then every different cycle of G_P is mapped into a different cycle of I_P. Hence, in this case, I_P has at least as many cycles as G_P has. On the other hand, it is not the case that every cycle in I_P corresponds to a cycle in G_P. Therefore, it is our expectation that in cases like that, the algorithm based on the rule graph will perform better than the algorithm based on the interaction graph.

5 Computational experience

The satisfiability, graph and linear programming methods described above were tested on randomly generated negative logic programs. Namely, a number of random directed graphs with N nodes and density d (density of a graph $G = (N, E)$ is the ratio $\| E \| / (\| N \|^2 - \| N \|)$) were generated as follows. For every pair of nodes (n_i, n_j) a random number $0 \leq n \leq 1$ is computed and if $n \leq d$ then the edge (n_i, n_j) is added to the graph. Every directed graph $G = (N, E)$ corresponds to the negative logic program $P = \{m \leftarrow \neg n_1 \wedge \neg n_2 \ldots \wedge \neg n_k | m \in N$ and for n_j, $1 \leq j \leq k$, $n_j \in \Gamma^-(m)\}$. Note that if P is a negative logic program then every kernel of G_P corresponds to a stable model of P. The same holds for the models of the propositional theory T_P. Consequently, the associated linear programming problem only consists of a set of inequalities (no objective function is present), which are to be solved.

We compared Zhang's SATO (Version 2.0) satisfiability algorithm ([13]), the graph algorithm which computes the kernels of a directed graph described in [9], and the integer linear programming CPLEX Mixed Integer Optimizer (Version 2.10). All the experiments were run on a SPARC-10. It was clear that the the linear programming method was remarkably inferior to the two other approaches. For example given a graph with 100 nodes, density 0.1 , and no kernel, SATO decided that the associated set of clauses is unsatisfiable in 0.1 sec, while CPLEX responded that there is no solution in 1369 seconds.

	.05		.10		.15		.20		.30		.40	
	SAT	GR	SAT	GR	SAT	GR	SAT	GR	SAT	GR	SAT	GR
80	0.02	0.07	0.05	0.21	0.09	0.36	0.13	0.38	0.18	0.34	0.35	0.30
90	0.04	0.12	0.08	0.44	0.19	0.95	0.21	0.83	0.31	0.65	0.36	0.45
100	0.04	0.16	0.14	1.10	0.24	1.45	0.29	1.53	0.48	1.12	0.51	0.72

Table 1

In Table 1 we present the run times (CPU time in seconds), of our experimentation with SATO (the SAT columns) and the graph algorithm (the GR columns). The time reported in each entry of the table is the average run time of 8 different problems, and it is the time taken by the implementations to compute all the stable models of the input logic programs. The first row shows the densities of the random graphs, while the first column the number of nodes.

It is clear that SATO performs better than the graph algorithm for graphs of densities less than 0.5, which we believe are interesting for the case of logic programs. However, we are currently working on a more efficient implementation of the graph algorithm, and the incorporation of methods for computing smaller feedback vertex sets.

6 Conclusions

In this paper we presented four different methods for computing the stable model semantics based on a graph representation of logic programs. We compared some of these methods both theoretically and empirically, and showed that the graph representation allows their effective use. We also presented a new Davis-Putman procedure suitable for logic programs, as well as an algorithm that computes a subset of the stable models of odd cycle free programs, with polynomial delay. Deriving an algorithm that combines the methods presented in this paper seems an interesting subject for further research.

The graph model remains valid in the case of disjunction free default theories ([12]). Therefore, the problem solving techniques presented here, can also be applied to the case of default logic.

Acknowledgements Many thanks to Peter Barth, Shiva Chaudhuri, David Plaisted, and Alberto Torres, for helpful discussions and comments.

References

1. C. Bell, A. Nerode, R. Ng, and V. S. Subrahmanian. Implementing deductive databases by linear programming. In *ACM, Principles of Database Systems*, 1992.
2. C. Bell, A. Nerode, R. Ng, and V. S. Subrahmanian. Implementing stable semantics by linear programming. In *Proc. of International Workshop on Logic Programming and Nonmonotonic Reasoning*, 1993.
3. R. Ben-Eliyahu and R. Dechter. Default logic, propositional logic and constraints. In *Proc. of AAAI-91*, pages 379–385, 1991.
4. R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. Technical Report 92-66, University of California, Irvine, 1992.
5. Chin-Liang Chang and Richard Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.
6. K. L. Clark. Negation as failure. In Gallaire and Minker, editors, *Logic and databases*, pages 293–322. Plenum Press, New York, 1978.
7. Y. Dimopoulos. Classical methods in nonmonotonic reasoning. Technical Report MPI-I-94-229, Max-Planck-Insitut für Informatik, 1994. forthcoming.
8. Y. Dimopoulos and V. Magirou. A graph-theoretic approach to default logic. To appear in *Information and Computation*, 1994.
9. Y. Dimopoulos, V. Magirou, and C. Papadimitriou. On kernels, defaults and even graphs. Technical Report MPI-I-93-226, Max-Planck-Institut für Informatik, 1993.
10. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.
11. C. Papadimitriou and M. Yannakakis. Tie-breaking semantics and structural totality. In *Proceedings Eleventh Symposium on Principles of Database Systems*, pages 16–22, 1992.
12. R. Reiter. A logic for default reasoning. *AI Journal*, 13:81–132, 1980.
13. H. Zhang. Sato: A decision procedure for propositional logic. *Association of Automated Reasoning Newsletters*, 22, 1993.

Partial Evaluation and Relevance for Approximations of the Stable Semantics

Jürgen Dix[1] and Martin Müller[2]

[1] University of Koblenz, Dept. of Computer Science,
Rheinau 1, D-56075 Koblenz, Germany
[2] Graduiertenkolleg Kognition and German Research Center for AI (DFKI),
Stuhlsatzenhausweg 3, D-66123 Saarbrücken

Abstract. There is no doubt that STABLE and WFS are among the dominant semantics for logic programs. While WFS has many nice structural properties, it is very weak. STABLE allows to derive more atoms, but may become inconsistent and it is not relevant, i.e. the truth value of an atom does not only depend on the call-graph below it. In this paper we consider the problem of defining approximations of STABLE that are both *relevant* and satisfy a general *partial deduction* property.

1 Introduction and Motivation

The well-founded semantics WFS [vGRS91] and the stable semantics STABLE [GL88] are undoubtedly the dominant semantics for normal logic programs. Both of them underly very simple intuitions: The well-founded *three-valued* model is in a sense the *weakest reasonable* semantics defined on *all* normal programs ([Dix92]). STABLE, on the other hand, is defined by a set of *two-valued* models which can therefore be used for both *sceptical* and *credulous* reasoning by considering *all* or *some* of the stable models for entailment. An additional, distinguishing feature of STABLE is that every stable model can be seen as a solution of the constraints expressed by the program. Such a solution corresponds to a complete (two-valued) scenario (the "scenario view" of a semantics [DM94]). The shortcomings of both semantics are also well-known: WFS is often weaker than intended since it prohibits case-based reasoning, and STABLE may become inconsistent. Moreover, the STABLE semantics does not allow top-down query processing, because the answer to a query does not only depend on its call-graph, i.e. the subprogram that can be recursively reached by tracing program clauses. Consider the well-known examples below: p is not in WFS(P_{split}), although $a \vee b$ and p hold in every two-valued model of P_{split}. STABLE(P_{split}) derives p while WFS(P_{split}) leaves p undefined.

Example 1 WFS vs. STABLE.

$$\begin{array}{llll}
P_{split}: & a \leftarrow \neg b & \qquad P_{no\text{-}stable}: & p \leftarrow \neg p \\
& b \leftarrow \neg a & & a \\
& p \leftarrow a & & \\
& p \leftarrow b & &
\end{array}$$

WFS behaves very regularly as indicated by many structural properties. In particular, it satisfies a very general principle of *partial evaluation* and it is *relevant*: Notions to be defined below. The well-founded model is an approximation of the sceptical semantics induced by the stable models. Some implementations of STABLE use this fact explicitly [SNV92] and some do not [CW93]. In this paper we consider the problem of closing the gap between WFS and STABLE by investigating extensions of WFS towards STABLE which still share the well-behavedness of WFS.

This goal is approached in two ways: The first singles out the *regular model semantics* REG-SEM [YY93] as such a well-behaved approximation. Since REG-SEM is given by a set of (three-valued) models, it still allows for scenario reasoning where the scenarios may become incomplete, though. The second one enriches the definition of STABLE by imposing certain properties. We develop a semantics STABLE^{rel} by trying to stick as close as possible with the spirit of (two-valued) STABLE and yet to avoid the inconsistency problem. STABLE^{rel} has a more constructive flavor.

The plan of the paper is as follows: In Section 2 below, we give some notation. In Section 3 we review the stable, the well-founded, and the regular semantics along with their relationship. Section 4 briefly describes the abstract properties we consider in this paper: *Cut*, *Relevance*, *PPE*, and *GPPE*. In Section 5 we use our approach to define a relevant version of STABLE: STABLE^{rel}. We end in Section 6 with some remarks about disjunctive logic programs, related attempts and conclude.

2 Notation

A program clause or a *rule* is a formula

$$a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots \neg c_l, \text{ where } n \geq 1 \text{ and } l \geq 0.$$

As usual, the comma represents conjunction. A *rule* is called *definite* if $l = 0$. a is said to be the *head* of the rule, $b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_l$ its *body*. A *program* P is a finite set of rules and it straightforwardly inherits the typology of clauses. The *Herbrand Base* induced by a program P is denoted B_P. In general the a, b_j, c_l are arbitrary atomic formulæ, but here we restrict ourselves to fully grounded programs. Furthermore, we spare to discuss infinite such programs by restriction to programs with the bounded term size property (e.g. *datalog* programs). In the sequel *program* will mean *fully grounded program* and its finiteness is to be understood.

We will also use the truth values $\mathbf{t}$, $\mathbf{u}$, and $\mathbf{f}$ which must be assigned the truth values *true*, *unknown*, or *false* in every model. We call $\mathbf{u}$-*programs* those programs that contain an occurrence of $\mathbf{u}$ in their bodies.

A program P induces a notion of *dependency* between atoms from B_P. We say that *a depends immediately* on b iff b appears in the body of a clause in P, such that a appears in its head. The two place relation *depends on* is the transitive closure of *depends immediately on*. The *dependencies of* and the *rules relevant for an atom x* are defined by:

- $\mathit{dependencies\_of}(x) := \{a : x \text{ depends on } a\}$,
- $\mathit{rel\_rul}(P, x)$ is the set of *relevant rules* of P with respect to x, i.e. the set of rules that contain an $a \in \mathit{dependencies\_of}(x)$ in their head,

These definitions are straightforwardly extended to arbitrary literals by setting $\mathit{dependencies\_of}(\neg x) := \mathit{dependencies\_of}(x)$ and $\mathit{rel\_rul}(P, \neg x) := \mathit{rel\_rul}(P, x)$. Our notion of a semantics is a very general one [Dix91, Dix92]:

Definition 1 SEM.
A semantics SEM is a mapping from the class of all programs into the powerset of the set of all three-valued Herbrand Structures. SEM assigns to every program P a set of three-valued Herbrand Models of P:

$$\mathrm{SEM}_P \subseteq \mathrm{MOD}_{3-val}^{Herb_{\mathcal{L}_P}}(P).$$

We associate with any semantics SEM a sceptical entailment relation SEM^{scept}.

Definition 2 Sceptical entailment relation $\mathrel{|\!\!\sim}_P$.
Let P be a program and U a set of atoms. Any semantics SEM induces a sceptical entailment relation SEM_P^{scept} as follows:

$$\mathrm{SEM}_P^{scept}(U) := \bigcap_{\mathcal{M} \in \mathrm{SEM}_P(U)} \{L : L \text{ is a pos. or neg. literal with: } \mathcal{M} \models L\}$$

As our main emphasis lies on the sceptical semantics, we will henceforth use SEM instead of SEM^{scept}. We distinguish two notions of one semantics being an *extension* of another one:

- $\mathrm{SEM} \leq_k \mathrm{SEM}'$: This means that SEM' classifies more atoms as true or false than SEM, and
- SEM' is defined for a class of programs that strictly includes the class of programs for which SEM is defined, and for all programs of this smaller class SEM and SEM' coincide.

As an example, STABLE extends WFS in the former sense (see Definition 4) while both STABLE and WFS extend the least Herbrand model semantics for *definite* or the least supported semantics for *stratified* programs in the latter sense. The notation $\leq_k$ comes from the partial ordering $\mathbf{u} \leq_k \mathbf{t}$ and $\mathbf{u} \leq_k \mathbf{f}$ (*knowledge* ordering, hence k). Minimal models are defined via the *truth* ordering $\leq_t$ defined by $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$.

3 Stable, Well-founded, and Regular Semantics

One of the central ideas of the stable semantics is that any atom in a potential model should have a definite reason to be true or false. This idea was made explicit by Bidoit and Froidevaux [BF91] and, independently, by Gelfond and Lifschitz [GL88]. We follow the presentation of [Prz90]:

Definition 3 Extended Gelfond-Lifschitz Transformation.
Let P be a program and $N \subseteq B_P$ be a three-valued model of P, then the *extended Gelfond-Lifschitz transformation* (or simply GL transformation) P^N is obtained from P by replacing every *negative* body literal $\neg c$ in P by **f** (resp. **u** or **t**) if N assigns c the truth value **t** (resp. **u** or **f**). Furthermore, a clause with **f** in its body can be erased, and **t** can be dropped from the body of every clause.

Note that P^N is always a *definite* **u**-program which has a unique $\leq_t$-minimal (three valued) Herbrand Model M_{P^N} ([Prz90, Proposition 3.1.]). Comparison of N and M_{P^N} answers whether N is a stable model:

Definition 4 STABLE, PART-STABLE, and REG-SEM.
N is called a *stable* model of P, if $M_{P^N} = N$ and N is two-valued. It is called a *partial stable* model if N is three-valued. The stable semantics STABLE(P) is given by the set of two-valued partial stable models, PART-STABLE(P) by the set of all, possibly three-valued, stable models of P. REG-SEM(P) is given by the set of $\leq_k$-maximal partial stable models of P: We call them *regular* models[3].

The well-founded semantics relates to STABLE as follows [vGRS91, Prz90]:

- WFS(P) coincides with PART-STABLE(P) (viewed sceptically).
- Every stable model N of P extends WFS(P): WFS(P) $\leq_k N$.
- If WFS(P) is two-valued, WFS(P) is the unique stable model.

The relationship between REG-SEM and a number of other semantics is discussed in [YY93] (see also [DM94]). Of course, WFS $\leq_k$ REG-SEM $\leq_k$ STABLE holds. For illustration consider the following example:

Example 2 REG-SEM vs. STABLE.

$$\begin{array}{rl@{\qquad}rl} P_{\neg\exists\, stab.}: & a \leftarrow \neg b & P_{regsem}: & a \leftarrow \neg b \\ & b \leftarrow \neg a & & b \leftarrow \neg a \\ & p \leftarrow \neg p & & p \leftarrow \neg a \\ & & & p \leftarrow \neg p \end{array}$$

$P_{\neg\exists stab.}$ has no stable model, but two regular ones: $\{a, \neg b\}$, and $\{b, \neg a\}$. P_{regsem} has the unique stable model $\{p, b, \neg a\}$, but it has two regular models, namely $\{p, b, \neg a\}$ and $\{a, \neg b\}$.

4 Abstract Approach to Reasonable Semantics

Many semantics have been defined in order to give meaning to logic programs with negation. The abstract approach to semantics has been introduced in [Dix92] in order to compare and evaluate semantics against each other and to avoid arguments based solely on small example programs. It has also been employed to reconstruct semantics based on intended structural properties. In this section we give the definition of only those principles which will be needed below.

[3] They were originally defined in [YY90].

4.1 Relevance and Cut

First consider the following two (closely related) properties, which we require any reasonable semantics SEM to have [Dix92]:

C_1: If $a \leftarrow l_1, \ldots, l_k \in P$ and SEM(P)$\models l_1 \wedge \ldots \wedge l_k$ then SEM(P)$\models a$, and
Cut: If SEM(P)$\models a$ and SEM$(P \cup \{a\}) \models x$ then SEM(P)$\models x$.

Most important for our present goal is the *Principle of Relevance*. For motivation, consider the development of a program as a set of implicational dependencies between literals. Whilst there may be mutual dependencies, the *basic structure* of a program is of a hierarchical nature which the programmer deliberately chooses: She will most probably write down the (recursive) definition of some literals before proceeding to others which may or may not use the program written so far. The *Principle of Relevance* states that the evaluation of a semantics has to respect these dependencies: The truthvalue of a literal l should only depend on the *relevant rules* of P with respect to l.

Definition 5 Relevance.
For all literals l: SEM$(P)(l)$ = SEM$(rel_rul(P,l))(l)$.

Relevant semantics, for instance WFS, allow *top-down query evaluation*. This has also been noted for an extension of WFS by Alferes et al. [ADP94]. Non-relevant semantics are not amenable to any form of top-down computation because they employ a global argument (a query Q may depend on atoms not in $dependencies\_of(Q)$). The latter applies to STABLE and is related to the nonexistence of stable models for programs such as $P_{no\text{-}stable}$ in Example 1: STABLE does not satisfy Relevance.

4.2 Variants of Partial Evaluation

Partial evaluation principles allow to transform a program into another one from which the semantics may be more efficiently computed, or which has a more regular structure. We consider two reasonable variants:

Definition 6 Strong PPE.
Let P be a program such that the atom c occurs only positively in P. Let $c \leftarrow rhs_1, \ldots, c \leftarrow rhs_n$ be all the rules of P with head c. Define P' to be a program obtained from P by replacing some $a \leftarrow c, body \in P$ by the rules

$$a \leftarrow rhs_1, body, \quad \ldots \quad a \leftarrow rhs_n, body$$

Note that the rules $c \leftarrow rhs_1, \ldots, c \leftarrow rhs_n$ are not removed (in contrast to the weak PPE [Dix92]). The *principle of partial evaluation* is: $SEM(P') = SEM(P)$.

Definition 7 GPPE.
The *generalized* principle of partial evaluation is obtained from PPE by dropping the assumption that c only occurs positively in P.

One central difference between PPE and GPPE is that stratifiability of a disjunctive program is invariant under PPE but not under GPPE. PPE is more syntax-sensitive so to speak. As an example for a semantics satisfying PPE but not GPPE, consider WFS$_C$ [Sch92] (or, equivalently, WFS$^+$ [Dix92]):

Example 3 Generalized PPE fails for WFS$_C$.

$$\begin{array}{llll} P_{split}: & a \leftarrow \neg b & P'_{split}: & a \leftarrow \neg b \\ & b \leftarrow \neg a & & b \leftarrow \neg a \\ & p \leftarrow a & & p \leftarrow \neg b \\ & p \leftarrow b & & p \leftarrow \neg a \end{array}$$

P'_{split} results from P_{split} by evaluating a (by $\neg b$) and evaluating b (by $\neg a$). WFS$_C$ derives p from P_{split}, but it does not derive p from P'_{split}. Note that if enforce GPPE (i.e., derive p from P'_{split} by declaration) then *Rationality* ([Dix91]) fails: WFS$_C(P'_{split})$ contains neither $\neg a$ nor $\neg b$. Yet, adding a and b does not only inhibit the derivation of p, but makes even $\neg p$ derivable!

REG-SEM derives p from P'_{split}. Indeed, GPPE holds for REG-SEM:

Theorem 8 GPPE holds for WFS, REG-SEM and STABLE.
WFS, REG-SEM and STABLE satisfy GPPE.

The fact that only *positive* occurrences of c are replaced, makes the proof obvious for WFS. For REG-SEM and STABLE, we observe that GPPE and the GL transformation commute, and that the following lemma holds:

Lemma 9 Positive u-programs are invariant under GPPE.
Let P be a positive **u***-program and P' the result of applying GPPE to P. Then P and P' have the same $\leq_t$-minimal models.*

5 STABLE^{rel}

In this section we develop STABLE^{rel} in order to approximate the twovalued stable semantics but to avoid the inconsistency problem. At the first glance, it might seem that the solution is a redefinition of STABLE in case no stable models exist. Recalling that *WFS $\leq_k$ STABLE* holds, we define:

$$\text{STABLE}^*(P) := \begin{cases} \text{WFS}(P), & \text{if no stable model exists,} \\ \text{STABLE}(P), & \text{otherwise,} \end{cases}$$

and observe that WFS $\leq_k$ STABLE* $\leq_k$ STABLE. This could be seen to be the semantics underlying the algorithm in [SNV92], where STABLE is computed but WFS remains as an approximation if there are no stable models. Already he program $P_{split} \cup \{p \leftarrow \neg p\}$ shows that STABLE* does not satisfy *Relevance*. But the following example illustrates that the inconsistency is obviously not the only problem.

Example 4 (Non-) Existence of Stable Models.

$$\begin{array}{llllll}
P_{\neg\exists\, stab.}: & a \leftarrow \neg b & P_{stable}: & a \leftarrow \neg b & P_{stable} \cup \{p\}: & a \leftarrow \neg b \\
 & b \leftarrow \neg a & & b \leftarrow \neg a & & b \leftarrow \neg a \\
 & p \leftarrow \neg p & & p \leftarrow \neg p & & p \leftarrow \neg p \\
 & & & p \leftarrow a & & p \leftarrow a \\
 & & & & & p
\end{array}$$

P_{stable} has the unique stable model $\{p, a\}$, but $P_{\neg\exists\, stab.}$ has no stable model. P_{stable} has the unique stable model $\{p, a\}$ and $P_{stable} \cup \{p\}$ has two stable models: $\{p, a\}$ and $\{p, b\}$.

From the single stable model of P_{stable} we derive a, but the relevant part of P with respect to a is just the program consisting of the first two clauses which have two stable models $\{a, \neg b\}$ and $\{\neg a, b\}$ and disallow the derivation of a.

We expect a reasonable semantics *not to derive a from P_{stable}*. Therefore, the semantics we are looking for cannot be an extension of STABLE*. Moreover:

Lemma 10. *There is no semantics SEM that coincides with STABLE, if stable models exist ($SEM^{scept}(P) = STABLE^{scept}(P)$), and satisfies* Relevance.

This lemma shows that the variant of the stable semantics recently defined by Sacca (the least undefined stable model from [Sac93]) is not relevant.

We now define a version of STABLE which is relevant. Reconsider Example 4 and use it positively. It suggests a natural approximation: We enforce *Relevance* by just *focussing on the relevant part* of a program in the redefinition of STABLE. This idea is captured by STABLE**:

$$\mathrm{STABLE}^{**}(P)(l) := \begin{cases} \mathrm{STABLE}(rel\_rul(P, l))(l), & \text{if stable models exist,} \\ \mathrm{WFS}(rel\_rul(P, l))(l), & \text{otherwise.} \end{cases}$$

Although this version does satisfy *Relevance*, it has another serious flaw:

Example 5 STABLE**.

$$\begin{array}{lll}
P_{STABLE^{**}}: & a \leftarrow \neg b & x \leftarrow \neg x, a \\
 & b \leftarrow \neg a & y \leftarrow \neg y, b \\
 & & w \leftarrow \neg x, \neg y.
\end{array}$$

Note that according to our definition of STABLE**, both $\neg x$ and $\neg y$ are derivable but w is not: The most natural condition C$_1$ (see Section 4) is not satisfied. Similar counterexamples show that *Cut* also does not hold.

The fall back to WFS may seem ad-hoc, but is the weakest possibility [Dix92]. The construction to follow, which repairs the failure of Relevance, does not really depend on WFS but can be parametrized by stronger semantics, too.

Example 5 indicates that we need a *recursive* definition which will yield STABLE^{rel}. First, we define an equivalence relation $\sim$ between atoms of B_P:

$$a \sim b \quad \underline{:\text{iff}} \quad (a \text{ depends on } b \text{ and } b \text{ depends on } a) \text{ or } a = b.$$

Second, we take $\preceq$ to be the partial order induced by $\sim$ on $\sim$-equivalence classes:

$$\overline{a} \preceq \overline{b} \quad \underline{:\text{iff}} \quad b \text{ depends on } a.$$

Call an atom to be *of order 0*, if $\overline{a}$ is minimal in $\preceq$. If, for some atom a, the maximal order of the atoms of which a depends is n, then a is called to be *of order* $n+1$. Since we only consider finite ground programs, all atoms get a finite order. We extend this notion to literals in the obvious way: A literal $\neg l$ is of order n iff l is.

Definition 11 STABLE^{rel}.
For literals l of order 0 we set

$$\mathrm{STABLE}_0^{rel}(P)(l) := \begin{cases} \mathrm{STABLE}(\mathit{rel\_rul}(P,l))(l), & \text{if stable models exist,} \\ \mathrm{WFS}(\mathit{rel\_rul}(P,l))(l), & \text{otherwise,} \end{cases}$$

and for literals l of order $n+1$

$$\mathrm{STABLE}_{n+1}^{rel}(P)(l) := \begin{cases} \mathrm{STABLE}(\mathit{rel\_rul}(P,l)^{STABLE_n^{rel}(P)})(l), & \text{if } \exists \text{ stab.mod.} \\ \mathrm{WFS}(\mathit{rel\_rul}(P,l)^{STABLE_n^{rel}(P)})(l), & \text{otherwise.} \end{cases}$$

That is, we reduce $\mathit{rel\_rul}(P,l))$ with all literals that have been evaluated to *true* or *false* in a previous step and then apply STABLE or WFS, respectively. According to STABLE^{rel} we now do derive w from $P_{STABLE^{**}}$ (Ex. 5).

The formal definition of the reduction $P^{\langle T;F \rangle}$ of a program P with respect to a three-valued model (see [Dix92]) is omitted due to space limitations. It is important to note that $P^{\langle T;F \rangle}$ is *not* the GL-transformation. It means that all occurrences of literals in P that are assigned a **t** or **f** by $\langle T;F \rangle$ are replaced by their truth-value and P is simplified accordingly (cf. Def. 3).

Note that if at some point in the computation, at order n, say, we are forced to switch to WFS because no stable models exist, this applies to all orders $m > n$. This follows as a corollary from the following lemma.

Lemma 12.
Let $P = P_1 \cup P_2$ such that for all $a \in B_{P_2}$: $\mathit{rel\_rul}(P,a) \subseteq P_2$. Then for all stable models $\mathcal{M}$ of P: $\mathcal{M}$ restricted to B_{P_2} is a stable model of P_2.

The proof is by induction on the definition of the Gelfond-Lifschitz transform.

Let us add some words to the problem of defining STABLE^{rel} for general programs, i.e. for infinite ground programs. Here we also have to consider literals of *infinite* order. This can be done by associating arbitrary ordinals to literals and defining STABLE$_\alpha^{rel}$ accordingly. Some literals will be assigned a limit ordinal then. Note that literals may not only depend on other literals of infinite order, but also on literals of arbitrarily high finite order. Take for example the program consisting of $P(f(x)) \leftarrow P(x)$ and $Q(a) \leftarrow P(x)$. $Q(a)$ depends on $P(f^n(a))$ for all n! Finally, observe that $\preceq$ is not well-founded. The program $P(x) \leftarrow P(f(x))$ together with a constant a gives an infinitely descending sequence $P(a), P(f(a)), \ldots$: This is an illustration that not every atom can be associated a *finite* order

Constructively, we feel that STABLE^{rel} captures the spirit of STABLE, while it may, for programs with a well-founded dependency graph, be computed along the structure of the program which seems to be close to the programmers intent.

Lemma 12 also shows that STABLE^{rel} is weaker than STABLE, and the more so the higher we ascend in the order of literals: This is, because not all stable models of clauses of order n or less are necessarily stable models of clauses of order $n+1$ or less. In particular we have

$$\text{WFS} \leq_k \text{STABLE}^{**} \leq_k \text{STABLE}^{rel} \leq_k \text{STABLE}.$$

In addition, STABLE^{rel} is well-behaved:

Lemma 13.
STABLE^{rel} satisfies GPPE and Relevance.

A final remark: It is possible to modify the definition of STABLE towards the scenario view by approximating models. In this case, one might compare the number of models of the subprogram $P^n \subseteq P$ of rules up to order n to those of $P^{n+1} \subseteq P$: If there are less models, then the definition of some literal of order $n+1$ must have ruled out one of the models. A contradiction even must have appeared at that level, where there are no more STABLE models at all, which can be returned to the programmer as an additional information.

6 Conclusion and Related Attempts

Consideration of disjunctive logic programs introduces more serious difficulties. It is easy to formulate GPPE for disjunctive programs: This has been done independently by Brass/Dix [BD94] and Sakama/Seki [SS94]. In both papers it is proved that DSTABLE (and GCWA) satisfy GPPE. However, DSTABLE inherits the failure of *Relevance* from its non-disjunctive version STABLE. We can now try to extend REG-SEM to disjunctive programs: Indeed, this can be done in a straightforward fashion. Unfortunately, GPPE does no longer hold for such a version DREG-SEM: Stefan Brass found a counterexample already for positive programs. But the semantics D-WFS constructed in [BD94] is relevant and satisfies GPPE (as well as PERFECT).

Concerning normal, i.e. non-disjunctive programs, we considered in a companion paper ([DM94]) the semantics WFS$_C$ and WFS$^+$ and showed how they relate to STABLE. Although they extend WFS, approximate STABLE and are relevant, they do not have the general partial deduction property (GPPE). In this paper we showed that REG-SEM is a better approximation. We also defined a semantics which is even closer to STABLE and still satisfies both properties. The overall construction of STABLE^{rel} is procedural in nature and seems to be better suited for Top-Down Query Evaluation techniques.

Let us finally note that the semantics of Minker and his group (DWFS, GDWFS and their extensions) do not even satisfy our weak version of PPE and therefore are not well-behaved ([Dix91]).

7 Acknowledgements

The authors would like to thank two anonymous referees for very helpful remarks. Their suggestions helped to improve the paper a lot.

References

[ADP94] J. J. Alferes, C. V. Damasio, and L. M. Pereira. Top-down Query Evaluation for Well-founded Semantics with Explicit Negation. *Proc. ECAI.* 1994.

[BD94] S. Brass and J. Dix. A Disjunctive Semantics based on Bottom-Up Evaluation and Unfolding. *13th World Computer Congress IFIP, GI-Workshop W2 (Disjunctive Logic Programming and Disjunctive Databases)*, 1994.

[CW93] W. Chen and D. S. Warren. Computation of Stable Models and its Integration with Logical Query Processing. Technical report, SUNY at Stony Brook, 1993.

[BF91] N. Bidoit and C. Froidevaux. General logical Databases and Programs: Default Logic Semantics and Stratification. *Information and Computation*, 91:15–54, 1991.

[Dix91] J. Dix. Classifying Semantics of Logic Programs. *Logic Programming and Non-Monotonic Reasoning, Proc. 1st Int. Workshop*, pp. 166–180. 1991.

[Dix92] J. Dix. A Framework for Representing and Characterizing Semantics of Logic Programs. *Principles of Knowledge Representation and Reasoning: Proc. 3rd Int. Conf. (KR '92)*, pp. 591–602. 1992.

[Dix94] J. Dix. A Classification-Theory of Semantics of Normal Logic Programs: II. Weak Properties. *Fundamenta Informaticae*, forthcoming, 1994.

[DM94] J. Dix and M. Müller. The Stable Semantics and its Variants: A Comparison of Recent Approaches. *Proc. 18th German Annual Conf. on Artificial Intelligence (KI '94).* Springer, LNAI, to appear, 1994.

[GL88] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. *5th Int. Conf. on Logic Programming*, pp. 1070-1080, 1988.

[Prz90] T. Przymusinski. Well-founded Semantics Coincides With Three-Valued Stable Semantics. *Fundamenta Informaticae*, XIII:445–463, 1990.

[Sac93] D. Sacca. The Expressive Power of Stable Models For DATALOG Queries with Negation. *Proc. Workshop on Logic Programming with Incomplete Information after ILPS' 93*, pp. 150–162, 1993.

[Sch92] J. S. Schlipf. Formalizing a Logic for Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 5:279–302, 1992.

[SNV92] V.S. Subrahmanian, Nau, and Vago. WFS + Branch and Bound = Stable Models. Technical Report CS-TR-2935, Univ. of Maryland, July 1992. To be published with *IEEE Transactions on Knowledge and Data Engineering*

[SS94] C. Sakama and H. Seki. Partial Deduction of Disjunctive Logic Programs: A Declarative Approach. *Proc. 4th Int. Workshop on Logic Program Synthesis and Transformation (LOPSTR'94).* Springer, LNCS, 1994.

[vGRS91] A. van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38:620–650, 1991.

[YY90] Jia-Huai You and Li-Yan Yuan. Three-valued Formalization of Logic Programming: is it needed. In *Proc. of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems,*, pages 172–182. ACM Press, 1990.

[YY93] J.-H. You and L.-Y. Yuan. On the Equivalence of Semantics for Normal Logic Programs. *Proc. Workshop on Logic Programming with Incomplete Information, following ILPS' 93*, pp. 161–176, 1993.

Circumscribing Features and Fluents: A Fluent Logic for Reasoning about Action and Change

Patrick Doherty[1] and Witold Łukaszewicz[2]

[1] Department of Computer and Information Science
Linköping University
S-58183 Linköping, Sweden
patdo@ida.liu.se
[2] Institute of Informatics
Warsaw University
00-913 Warsaw 59, Poland

Abstract. Sandewall has recently proposed a systematic approach to the representation of knowledge about dynamical systems that includes a general framework in which to assess the range of applicability of existing and new logics for action and change. As part of the framework, several logics of preferential entailment are introduced and assessed for particular classes of action scenario descriptions. The intent of this paper is to provide syntactic characterizations of several of these relations of preferential entailment in terms of circumscription with a standard base logic consisting of FOPC with temporal terms and discrete time. It turns out that *occluded circumscription*, which covers the broadest class of action scenarios, and includes many of the most problematic scenarios studied in the literature, is one of the most straightforward logics considered. The class includes scenarios with non-deterministic actions, actions with duration, partial specification of any state including the first, and incomplete specification of the timing and order of actions.

1 Introduction

Sandewall has recently proposed a systematic approach to the representation of knowledge about dynamical systems that provides a general framework which can be used to assess the range of applicability of existing and new logics for action and change. The assessment is based on making explicit both the ontological and epistemological assumptions about the world and agents acting in it. One can then prove proposed logics *sound* and *complete* for various classes of action scenario descriptions. Recent work in the area reinforces the importance of such an approach where the assessment of logics is less dependent on representative examples and is instead studied in terms of particular classes of action scenarios ([5],[6]). Details of Sandewall's work may be found in several papers ([8],[10]), and more recently, in a book manuscript [9].

In this paper, we present a logic of fluents (FL) to represent action scenario descriptions and provide syntactic characterizations of a number of relations of preferential entailment used by Sandewall. Each entailment relation is presented in terms of circumscriptive axioms, rather than the model theoretic presentation Sandewall uses which is based on a non-standard temporal feature logic. The benefits are twofold. Firstly, the syntactic characterizations in terms of circumscription can be used as a basis for implementation. Secondly, the circumscriptive representations can more easily be used as a basis for comparing Sandewall's work with numerous existing logics of action and change formulated in in terms of circumscription.

1.1 Action Scenario Descriptions

A majority of reasoning problems involving action and change can be represented in terms of *(action) scenario descriptions* . A scenario description is a partial specification in a logical language of the initial state of a system, combined with descriptions of some of the actions that have occurred together with their timing. The "Yale shooting problem" or "Stanford murder mystery" may be considered scenario descriptions.

The formal syntax for specifying scenario descriptions is defined in terms of a surface language $\mathcal{L}(SD)$, consisting of action occurrence statements (ac-), action (law) schemas (acs-), and observation statements (obs-). In what follows, all expressions occurring in scenario descriptions will be prefixed with the symbols "obs", "ac" and "acs" to denote the appropriate type of statement.

Example 1. The following is the Yale shooting scenario (below a and l are fluent constants standing for *alive* and *loaded*, respectively, while *Load* and *Fire* are action symbols).

obs1 $[0]\ a \wedge \neg l$
ac1 $[2,4]\ Load$
ac2 $[5,6]\ Fire$
acs1 $[t_1, t_2]\ Load \rightsquigarrow [t_1, t_2]\ l := T$
acs2 $[t_1, t_2]\ Fire \rightsquigarrow ([t_1]\ l \rightarrow [t_1,t_2](\neg a := T \wedge \neg l := T)$.

In the initial state, the gun is not loaded and the turkey is alive. The gun is loaded during the period 2 to 4, and fired during the period 5 to 6. In the interim between the two actions is a waiting period. The action schema for "Load" states that if a gun is loaded during the period t_1 to t_2, then at the end of the duration the gun will in fact be loaded at time t_2. A similar type of reading may be given to the action schema for "Fire". Note that during the duration of an action, the values of influenced fluents are not known. Expressions of type $[t, t']\gamma := T$ are used to assert that a fluent is reassigned a new value of *true* or *false*. The idea is that during the duration t to t', we do not know when γ becomes *true* or *false*, but only that it is *true* or *false* at time t'.

The surface syntax of a scenario description is translated into wffs in $\mathcal{L}(FL)$, the language for the base logic FL, using a two step process. In the first step, a scenario description is translated into a chronicle description by expanding the action schemas in the scenario. The corresponding chronicle description for the YSP is,

obs1 [0] $a \wedge \neg l$
scd1 [2,4] $l := T$
scd2 [5] $l \rightarrow$ [5,6]$(\neg a := T \wedge \neg l := T)$.

1.2 The Reasoning Task

After translating a chronicle description into a set of wffs in $\mathcal{L}(FL)$, the reasoning task is to derive additional statements about the course of events. Temporal prediction and postdiction, among others, are two types of reasoning tasks one might be interested in. In a sense, this distinction is unimportant. Essentially, what we would like to do is to conclude as much as possible about the values of fluents at different points in time. Such sets of conclusions will be referred to as *chronicle completions.* Of course, our interest is in intended completions, i.e. those completions containing desirable conclusions constrained by an assumption of inertia. For instance, the Yale shooting chronicle has exactly one intended completion given by

[0,2] $a \wedge \neg l$
[3] a
[4,5] $a \wedge l$
[6,∞) $\neg a \wedge \neg l$

Within Sandewall's framework, the intent is to find the proper relation of preferential entailment for the class that this scenario description belongs to, and show that for any scenario in this class, only the conclusions in the intended completions for each scenario description are in fact preferentially entailed. Although this is the basic methodology, we will appeal to representative examples in this paper as a vehicle for presenting the various logics.

2 Fluent Logic

In this section, we provide a summary of the relevant parts of the fluent logic (FL).

- Assume FOPC with equality and with quantification only over temporal variables. Temporal terms include constants for each natural number, temporal variables $s, s', \ldots, t, t', \ldots$, and $pred(\mathbf{t})$, where $pred$ is a function, $\mathbf{t}$ is a temporal term and $pred(\mathbf{t})$ is $\mathbf{t} - 1$.
- Atomic formulas are of two types:
  - $fl(\mathbf{t})$, where $\mathbf{t}$ is a temporal term and fl is a fluent symbol.
  - $\mathbf{t} = \mathbf{t}'$, $\mathbf{t} < \mathbf{t}'$, $\mathbf{t} \leq \mathbf{t}'$, where $\mathbf{t}$ and $\mathbf{t}'$ are temporal terms and $=$, $<$, and $\leq$ are binary predicate symbols.
- The set of formulas is constructed in the usual way with the usual logical connectives.
- Certain abbreviations are used in chronicle descriptions to provide a clear and succinct representation, but can always be compiled to formulas in $\mathcal{L}(FL)$. Here are a few relevant abbreviations:[3]

[3] Here at, $at1$ and $at2$ are fluent symbols or their negations, whereas δ stands for a *semi-formula* that is either a fluent symbol or a propositional combination of fluent

$$[\mathbf{t}]\delta \overset{\text{def}}{=} \delta(\mathbf{t})$$
$$[\mathbf{s},\mathbf{t}]\delta \overset{\text{def}}{=} \forall t.\ \mathbf{s} \leq t \leq \mathbf{t} \rightarrow \delta(t)$$
$$[\mathbf{s},\mathbf{t}]at := T \overset{\text{def}}{=} \exists t.\mathbf{s} \leq t < \mathbf{t} \wedge \forall t'(t < t' \leq \mathbf{t} \rightarrow at(t'))$$
$$[\mathbf{s},\mathbf{t}](at1 := T \vee at2 := T) \overset{\text{def}}{=} [\mathbf{s},\mathbf{t}]at1 := T \vee [\mathbf{s},\mathbf{t}]at2 := T$$
$$[\mathbf{s},\mathbf{t}](at1 := T \wedge at2 := T) \overset{\text{def}}{=} [\mathbf{s},\mathbf{t}]at1 := T \wedge [\mathbf{s},\mathbf{t}]at2 := T.$$

Definition 1 Interpretation. An *interpretation* for FL is a pair M=⟨N, m⟩, where N is the set of all natural numbers and m is a function which assigns to a pair consisting of a fluent symbol and a natural number, a truth-value from the set {T,F}.

To represent actions, we extend the alphabet of FL by introducing a set of *action symbols.* These will usually be denoted by everyday English words such as *Load*, *Fire*, etc. An *action* is any expression of the form [s,t]**A**, where **A** is an action symbol and s and t are temporal terms. An *action schema* is any expression of the form $[s,t]\mathbf{A} \leadsto \alpha$, where **A** is an action symbol, *s* and *t* are temporal variables and α is any formula of $\mathcal{L}(FL)$. Scd- statements are instantiated to the right side of the corresponding acs- statements resulting in chronicle descriptions.

3 Chronological Circumscription

In this section, we introduce chronological circumscription, mainly as a vehicle for defining filtered circumscription in the next section.

Let $\bar{\Phi} = (\Phi_1, \ldots \Phi_n)$ and $\overline{fl} = (fl_1, \ldots fl_n)$ be tuples of predicate variables and fluent symbols, respectively. We write $\bar{\Phi} < \overline{fl}$ to denote the formula

$$\exists t \forall t'. \left\{ \left[\bigwedge_{i=1}^{n} [t' < t \rightarrow \Phi_i(t') \equiv fl_i(t')] \right] \wedge \left[\bigvee_{i=1}^{n} \Phi_i(t) \not\equiv fl_i(t) \right] \wedge \right.$$
$$\left. \bigwedge_{i=1}^{n} [[\Phi_i(t) \not\equiv \Phi_i(pred(t))] \rightarrow [fl_i(t) \not\equiv fl_i(pred(t))]] \right\}$$

$\bar{\Phi} < \overline{fl}$ states that there is a time point, t, such that Φ_i's and fl_i's are identical on all earlier points, at least one Φ_i differs from fl_i on t, and, if some Φ_i changes its truth value on t, then so does fl_i. In the sequel we will use C both for the chronicle and the formula which is the conjunction of the members of C.

symbols. Suppose that t is a temporal term. We write $\delta(\mathbf{t})$ to denote the formula of FL obtainable by replacing each fluent symbol fl occurring in δ by $fl(\mathbf{t})$. For example, if δ is $loaded \rightarrow alive$, then $\delta(4)$ is $loaded(4) \rightarrow alive(4)$.

Definition 2 Chronological Circumscription. Let $\overline{fl} = (fl_1, \ldots, fl_n)$ be the tuple of all fluent symbols occurring in a chronicle C. The *chronological circumscription of* $\overline{fl}$ in C, written $\mathrm{CIRC_{CHR}}(C)$, is the sentence

$$C(\overline{fl}) \wedge \forall \bar{\Phi}. \neg[C(\bar{\Phi}) \wedge \bar{\Phi} < \bar{fl}]$$

where $\bar{\Phi} = (\Phi_1, \ldots, \Phi_n)$ is a tuple of predicate variables.

A formula α is said to be a *consequence* of chronological circumscription of $\overline{fl}$ in C iff $\mathrm{CIRC_{CHR}}(C) \models \alpha$.

We now provide a semantics for chronological circumscription. Let M=$\langle$N,m$\rangle$ be an interpretation. The *breakset* of M at a time **i** is specified as breakset(M, **i**) = $\{fl \mid \mathrm{m}(fl, \mathbf{i}-\mathbf{1}) \neq \mathrm{m}(fl, \mathbf{i})\}$. Intuitively, the breakset at a time point is the set of fluents that changed their truth value at that point.

Definition 3. Let M=$\langle$N,m$\rangle$ and M'=$\langle$N,m'$\rangle$ be interpretations. We say that M is *chronologically less than* M', written $\mathrm{M} <_{\mathrm{CHR}} \mathrm{M'}$, iff there exists a natural number **i** such that

- m(fl, **j**)= m'(fl, **j**), for each fluent symbol fl and each **j** less than **i**;
- breakset(M,**i**)$\subset$ breakset(M',**i**).

An interpretation M is said to be *chronologically minimal* in a class of interpretations $\mathcal{I}$, if there is no interpretation N from $\mathcal{I}$ such that $\mathrm{N} <_{\mathrm{CHR}} \mathrm{M}$.

For any set of formulas X, we write MOD(X) to denote the class of all models of X. By $\mathrm{MOD^{CHR}}(X)$ we denote the class of all interpretations chronologically minimal in MOD(X).

Theorem 4. For any chronicle C, $\mathrm{MOD}(\mathrm{CIRC_{CHR}}(C)) = \mathrm{MOD^{CHR}}(C)$.

4 Filtered Circumscription

The use of chronological minimization, in any form, has been shown to be somewhat restrictive. Most importantly, it is inappropriate for the reasoning task of temporal postdiction, where we reason from known facts at later points in time to inferred facts at earlier points in time. Sandewall's solution to this problem is simple but elegant [8]. Rather than chronologically minimizing all formulas in a chronicle description, one takes advantage of the partitioning of chronicle formulas into observations (`obs-`) and schedule statements (`scd-`). One first applies chronological minimization to the schedule statements which result in a set of potential histories minimized for change and then filters this set by intersecting the associated models with the models for the observation statements. Kartha and Lifschitz [4] have recently proposed the use of both filtering and occlusion (releases), to deal with simple forms of non-determinism.

For simplicity, assume that each chronicle description C is a pair $\langle OBS, SCD\rangle$, where OBS and SCD are the observation axioms and the schedule axioms of C.

Definition 5 Filter Chronological Circumscription. Assume that $\overline{fl} = (fl_1, \ldots, fl_n)$ is the tuple of all fluent symbols occurring in a chronicle $C{=}\langle OBS, SCD\rangle$. The *filter chronological circumscription of* $\overline{fl}$ in C, written $\mathrm{CIRC}_{\mathrm{FCHR}}(C)$, is the sentence $C \wedge \mathrm{CIRC}_{\mathrm{CHR}}(SCD)$.

A formula α is said to be a *consequence* of filter chronological circumscription of $\overline{fl}$ in C iff $\mathrm{CIRC}_{\mathrm{FCHR}}(C) \models \alpha$.

The semantics of filter chronological circumscription follows easily from that of chronological circumscription.[4]

Theorem 6. For any chronicle $C = \langle OBS, SCD\rangle$, $\mathrm{MOD}(\mathrm{CIRC}_{\mathrm{FCHR}}(C))= \mathrm{MOD}^{\mathrm{CHR}}(SCD) \cap \mathrm{MOD}(C)$.

Example 2 Stanford Murder Mystery. Consider the following scenario description,[5]

obs1 $[0]\ a$
ac1 $[2,3]\ Fire$
acs1 $[t_1,t_2]\ Fire \leadsto ([t_1]\ l \longrightarrow [t_1,t_2](\neg a := T \wedge \neg l := T)$
obs2 $[4]\ \neg a$

and the resulting chronicle description:

obs1 $[0]\ a$
scd1 $[2]\ l {\longrightarrow} [2,3](\neg a := T \wedge \neg l := T)$
obs2 $[4]\ \neg a$.

The intended chronicle completion is,

$[0,2]\ a \wedge l$
$[3,\infty)\ \neg a \wedge \neg l$

The scd1 axiom of the chronicle has four chronologically minimal models shown in Table 1. Clearly $\{\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3, \mathrm{M}_4\} \cap \mathrm{MOD}(C) = \mathrm{M}_1$, which is the model of the intended completion of SMMC.

5 Occluded Circumscription

There are two related problems with chronological circumscription: 1) it prefers fluent value changes as late as possible, and 2) as a side effect, when a change occurs in an interval, we are unable to remain agnostic about when the change actually takes place; it will occur as late as possible in the interval. This causes problems for actions with duration and randomized effects of actions, among other things.

To deal with such problems, Sandewall introduces an occlusion predicate on fluents, where an occluded fluent is exempt from the global persistence assumption. Consequently, where it would normally keep its value, when occluded it

[4] Note that for technical reasons, we intersect the preferred models of SCD with those of the whole chronicle C, rather than with just OBS. This has no effect on intended completions.

[5] The fluent symbols used below have the same meaning as in the Yale shooting scenario.

may vary. This is similar to varying predicates in ordinary circumscriptive logics, but not quite the same because the fluent may be occluded at some time points but not others.

Some minor adjustments need to be made to both our language and interpretations to incorporate occlusion. For each fluent symbol fl in the alphabet, we introduce a new symbol fl^*. The intuition is that when fl^* is true at a timepoint t then fl is occluded at t. We also redefine an interpretation as a triple $\mathrm{M} = \langle \mathrm{N}, \mathrm{m}, \mathrm{m}* \rangle$, where N is the set of natural numbers, and m (resp. m*) is a function which assigns to a pair consisting of a fluent symbol (resp. an occlusion fluent symbol) and a natural number, a truth value from {T,F}.

The occlusion technique amounts to the assumption that fluent symbols changing their truth-values during an action are occluded during the interval in which the action takes place. This is technically achieved by replacing any formula of the form $[t_1, t_2]fl := T$ (resp. $[t_1, t_2]\neg fl := T$) occurring in the expansion of an action schema by $[t_1, t_2]fl := T \wedge (t_1, t2]fl^*$ (resp. $[t_1, t_2]\neg fl := T \wedge (t_1, t2]fl^*$). For instance, the result of the action [4,6] *Load* wrt the action schema $[t_1, t_2]$ *Load* $\leadsto [t_1, t_2]\ l := T$ is the formula $[4,6]\ l := T \wedge (4,6]\ l^*$.[6]

In addition, we will need a set of *nochange* axioms which state that a fluent can only change its value at a timepoint if it is occluded at that timepoint. Let C be a chronicle and suppose that FLS is the set of all fluent symbols occurring in C. The set of *nochange axioms for* C, written NCH(C), is specified by

$$\mathrm{NCH}(C) = \{\forall t.[fl(t) \not\equiv fl(pred(t)) \rightarrow fl^*(t)] \mid fl \in \mathrm{FLS}\}.$$

The basic idea of occluded circumscription is to minimize occlusion in the schedule axioms of a given chronicle $C = \langle OBS, SCD \rangle$, and then to intersect the obtained models with those of OBS and NCH(C). Note that this technique reduces the model set both syntactically, by incrementing the axiom set with additional NCH axioms, in addition to the traditional preferred model approach. We now consider the details of occluded circumscription.

If X and Y are truth values from {T,F}, then we write X<Y iff X=F and Y=T.

Definition 7. Let $\mathrm{M} = \langle \mathrm{N}, \mathrm{m}, \mathrm{m}* \rangle$ and $\mathrm{M}' = \langle \mathrm{N}, \mathrm{m}', \mathrm{m}'* \rangle$ be interpretations. We say that M is *less than or equal to* M' *wrt occlusion*, written $\mathrm{M} \leq_{OCL} \mathrm{M}'$, iff for each fluent symbol fl and each natural number **i**

- $\mathrm{m}(fl,\mathrm{i}) = \mathrm{m}'(fl,\mathrm{i})$
- $\mathrm{m}^*(fl^*,\mathrm{i}) = \mathrm{m}'^*(fl^*,\mathrm{i})$ or $\mathrm{m}^*(fl^*,\mathrm{i}) < \mathrm{m}'^*(fl^*,\mathrm{i})$.

We write $\mathrm{M} <_{OCL} \mathrm{M}'$ iff $\mathrm{M} \leq_{OCL} \mathrm{M}'$, but not $\mathrm{M}' \leq_{OCL} \mathrm{M}$. An interpretation M is said to be *occlusion minimal* in a class of interpretations $\mathcal{I}$, if there is no interpretation M' from $\mathcal{I}$ such that $\mathrm{M}' <_{OCL} \mathrm{M}$.

[6] We use the form $(4,6]l^*$ rather than $[4,6]l^*$ because the intention is that l^* is true at 5 if l is occluded between the points $pred(5)$ and 5.

For any set of formulas X, we write $\mathrm{MOD}^{\mathrm{OCCL}}(X)$ to denote the class of all interpretations occlusion minimal in $\mathrm{MOD}(X)$.

Definition 8 Occlusion preferred models. Let $C{=}\langle OBS, SCD\rangle$ be a chronicle. We define the class of *occlusion preferred* models of C as

$$\mathrm{MOD}^{\mathrm{OCCL}}(SCD) \cap \mathrm{MOD}(OBS) \cap \mathrm{MOD}(\mathrm{NCH}(C)).$$

We now proceed to formulate a circumscription axiom capturing the occlusion preferential entailment. In what follows, by a *simple predicate expression* we understand a predicate variable, a fluent symbol or an occlusion fluent symbol.

Let $\bar{U} = (U_1, \ldots, U_n)$ and $\bar{V} = (V_1, \ldots, V_n)$ be tuples of simple predicate expressions. We write $\bar{U} \leq \bar{V}$ and $\bar{U} = \bar{V}$ to denote the formulas

$$\bigwedge_{i=1}^{n} [\forall t. U_i(t) \rightarrow V_i(t)] \text{ and } \bigwedge_{i=1}^{n} [\forall t. U_i(t) \equiv V_i(t)],$$

respectively. We write $\bar{U} < \bar{V}$ as an abbreviation for $\bar{U} \leq \bar{V} \wedge \neg(\bar{U} = \bar{V})$.

$\bar{U} < \bar{V}$ expresses that the extension of each member of $\bar{U}$ is a subset of the extension of the corresponding member of $\bar{V}$, and at least one of them is a proper subset.

Definition 9. [Occluded Circumscription] Let $C{=}\langle OBS, SCD\rangle$ be a chronicle and suppose that $\overline{fl} = (fl_1, \ldots, fl_n)$ and $\overline{fl}^* = (fl_1^*, \ldots, fl_n^*)$ are tuples of all fluent symbols and all occlusion fluent symbols occurring in C. *Occluded circumscription* of $\overline{fl}, \overline{fl}^*$ in C, written $\mathrm{CIRC}_{\mathrm{OCCL}}(C)$, is the sentence

$$\mathrm{NCH}(C) \wedge OBS \wedge \forall \bar{\Phi} \bar{\Psi}. \neg[SCD(\bar{\Phi}, \bar{\Psi}) \wedge \bar{\Phi} = \overline{fl} \wedge \bar{\Psi} < \overline{fl}^*]$$

where $\bar{\Phi} = (\Phi_1, \ldots \Phi_n)$, $\bar{\Psi} = (\Psi_1, \ldots \Psi_n)$ are tuples of predicate variables and $SCD(\bar{\Phi}, \bar{\Psi})$ is obtained from SCD by replacing each fluent symbol fl_i by Φ_i and each occlusion fluent symbol fl_i^* by Ψ_i.

The next theorem shows that occluded circumscription indeed captures the occlusion preferential entailment.

Theorem 10. For any chronicle $C = \langle OBS, SCD\rangle$,

$$\mathrm{MOD}(\mathrm{CIRC}_{\mathrm{OCCL}}(C)) = \mathrm{MOD}^{\mathrm{OCCL}}(SCD) \cap \mathrm{MOD}(OBS) \cap \mathrm{MOD}(\mathrm{NCH}(C)).$$

Example 3 Russian Turkey Scenario. The world of this scenario is the same as for the Yale shooting scenario, but a new action for spinning the gun's chamber is added. The effect of this action is that randomly the gun may or may not be loaded after its execution, regardless of whether it was loaded before or not. The scenario is represented as follows.

obs1 $[0]\ a \wedge \neg l$
ac1 $[1,2]\ \mathit{Load}$
ac2 $[3,4]\ \mathit{Spin}$
ac3 $[5,6]\ \mathit{Fire}$
acs1 $[t_1,t_2]\ \mathit{Load} \rightsquigarrow [t_1,t_2]\ l := T$
acs2 $[t_1,t_2]\ \mathit{Spin} \rightsquigarrow [t_1,t_2](\ l := T \vee \neg l := T)$
acs3 $[t_1,t_2]\ \mathit{Fire} \rightsquigarrow ([t_1]\ l \!\rightarrow [t_1,t_2](\neg a := T \wedge \neg l := T))$

This scenario leads to the following chronicle:

obs1 $[0]\ a \wedge \neg l$
scd1 $[1,2]\ l := T$
scd2 $[3,4]\ (l := T \vee \neg l := T)$
scd3 $[5]\ l \!\rightarrow\! [5,6]\ (\neg a := T \wedge \neg l := T)$

This chronicle has two intended completions. One where the gun becomes unloaded after spinning the chamber, and firing leads to no change, and one where spinning leads to no change, and after firing the person becomes dead; so given the above chronicle we want to remain agnostic as to whether the person is alive or not after the action *Fire* has been executed. Unfortunately, both chronological and filter chronological circumscription ignore the first of the intended completions and allow one to conclude that the person is dead after firing.

The modified chronicle, with the additional occluded fluents, corresponding to the Russian Turkey Scenario is,

obs1 $[0]\ a \wedge \neg l$
scd1 $[1,2]\ l{:=}\mathrm{T} \wedge$
$(1,2]\ l^*$
scd2 $[3,4](l{:=}\mathrm{T} \vee \neg l{:=}\mathrm{T}) \wedge$
$(3,4]\ l^*$
scd3 $[5]\ l \!\rightarrow [5,6](\neg a := T \wedge \neg l := T) \wedge$
$(5,6]\ a^* \wedge l^*$

There are two occlusion preferred models for this chronicle description shown in Table 2. Obviously, these are the intended models of the Russian turkey scenario.

6 Conclusions

Occluded circumscription, using both filtering and occlusion, provides a correct notion of entailment for a very broad class of action scenarios, under an assumption of simple inertia. In related papers ([1], [2]), it is shown that occluded circumscription, can in fact be reduced to a monotonic first-order formalism, opening up the possibility for efficient implementation of the class of scenarios covered by this logic. Due to the similarity between explanation closure and nochange, comparison with both Reiter [7] and Schubert [11] would be worthwhile, along with the recent work by Gelfond and Lifshitz [3].

Tables

Tab. 1: Stanford Murder Mystery

	M_1	M_2	M_3	M_4
0	a, l	$a,\neg l$	$\neg a,l$	$\neg a, \neg l$
1	a, l	$a,\neg l$	$\neg a,l$	$\neg a, \neg l$
2	a, l	$a,\neg l$	$\neg a,l$	$\neg a, \neg l$
3	$\neg a, \neg l$	$a,\neg l$	$\neg a,\neg l$	$\neg a, \neg l$
4	$\neg a, \neg l$	$a,\neg l$	$\neg a,\neg l$	$\neg a, \neg l$
⋮	$\neg a, \neg l$	$a,\neg l$	$\neg a,\neg l$	$\neg a, \neg l$

Tab. 2: Russian Turkey Shoot

	M_1	M_2
0	$a, \neg l,\neg a^*,\neg l^*$	$a, \neg l,\neg a^*,\neg l^*$
1	$a, \neg l,\neg a^*,\neg l^*$	$a, \neg l,\neg a^*,\neg l^*$
2	$a, l,\neg a^*,l^*$	$a, l,\neg a^*,l^*$
3	$a, l,\neg a^*,\neg l^*$	$a, l,\neg a^*,\neg l^*$
4	$a, l,\neg a^*,l^*$	$a, \neg l,\neg a^*,l^*$
5	$a, l,\neg a^*,\neg l^*$	$a, \neg l,\neg a^*,\neg l^*$
6	$\neg a, \neg l,a^*,l^*$	$a, \neg l,a^*,l^*$
7	$\neg a, \neg l,\neg a^*,\neg l^*$	$a, \neg l,\neg a^*,\neg l^*$
⋮	$\neg a, \neg l,\neg a^*,\neg l^*$	$a, \neg l,\neg a^*,\neg l^*$

References

1. P. Doherty. Reasoning about action and change using occlusion. In *Proceedings of the 11th European Conference on Artificial Intelligence, Aug. 8-12, Amsterdam,* pages 401–405, 1994.
2. P. Doherty and W. Łukaszewicz. Circumscribing features and fluents. In *Proceedings of the 1st International Conference on Temporal Logic*, 1994.
3. M. Gelfond and V. Lifschitz. Representing actions in extended logic programming. In K. Apt, editor, *Proc. Joint Int'l Conf. and Symp. on Logic Programming,* pages 559–573, 1992.
4. G. N. Kartha and V. Lifschitz. Actions with indirect effects (preliminary report). In *Proc. of the 4th Int'l Conf. on Principles of Knowledge Representation and Reasoning, (KR-94),* pages 341–350, 1994.
5. V. Lifschitz. Toward a metatheory of action. In *Proc. of the 2nd Int'l Conf. on Knowledge Representation and Reasoning, (KR-91)*, pages 376–386, 1991.
6. F. Lin and Y. Shoham. Provably correct theories of action (preliminary report). In *National Conference on Artificial Intelligence (AAAI-91),* pages 349–354, 1991.
7. R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.
8. E. Sandewall. Filter preferential entailment for the logic of action and change. In *Proc. Int'l Joint Conf. on Artificial Intelligence, (IJCAI-89)*, 1989.
9. E. Sandewall. The range of applicability of nonmonotonic logics for the inertia problem. In *Proc. Int'l Joint Conf. on Artificial Intelligence, (IJCAI-93)*, 1993.
10. E. Sandewall. Features and fluents: A systematic approach to the representation of knowledge about dynamical systems. Technical report, Department of Computer and Information Science, Linköping University, 1994. Final Review Version.
11. L. Schubert. Monotonic solution of the frame problem in situation calculus. In H. E. Kyburg, R. P. Loui, and G. N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.

Paraconsistency and Beyond: A New Approach to Inconsistency Handling

Suryanil Ghosh

Dept. of Computing Science, University of Alberta,
Edmonton, Alberta, Canada - T6G 2H1
email: ghosh@cs.ualberta.ca

Abstract. The expressive power of positive logic programs is enhanced by explicit negation by providing a natural and unambiguous way to assert negated information. But by this, we are faced with the problem of dealing with contradictions in the database. The ECSQ (for 'ex contradictione sequitir quodlibet') approach of classical logic, by which everything follows from a contradiction, thus resulting in the collapsing (or trivialization) of the system is certainly not a pragmatic approach towards handling inconsistency. We propose an inconsistency handling concept - **explicit paraconsistency**, formalized as **Approach $\mathcal{C}$**, which is close in spirit to 'paraconsistent' approaches known from the logico-philosophical literature on non-classical logics handling inconsistency. Furthermore, our approach goes beyond the paraconsistent horizon i.e. allowing nontrivial reasoning in presence of inconsistency. We allow reasoning from inconsistent information, keep track of conclusions inferred from inconsistent premises and propagate a new horizon of reasoning involving elements 'affected' by inconsistency. This we call **reasoning beyond paraconsistency** and formalize as **Approach $\mathcal{C}_d$**.

1 Introduction

Our objective in this work is to propose a pragmatic approach for treating knowledge in the presence of contradiction. In particular this approach would, instead of providing automatic belief revision or trivialization (as in classical logic), suggest a way out of the dilemma imposed by the presence of the ghost of contradiction. This capacity of considering inconsistent knowledge as being as relevant as any other knowledge, capturing it explicitly, reporting it and producing from this new interesting knowledge is what we define as *explicit paraconsistency* and *reasoning beyond paraconsistency.*

This cannot be attained via modal treatments, nonmonotonic logics or truth-maintenance devices and only approximately via many-valued logics because such approaches work by preventing contradictions or by remedying the situations instantly if inconsistency occurs, while we are interested in 'coexisting' with the contradiction, possibly taking profit of it as interesting knowledge, in the sense that understanding the causes of contradictions provides additional knowledge and also in the sense that contradiction driven reasoning provides a new horizon of new information which would have never been arrived at following the existing approaches of handling inconsistency. This allows us a wider spectrum of information, some irrefutable, some defeasible (the ones inferred from assumptions) and the new ones, contradictory and contradiction affected, which have different epistemicity than the others. It is a matter of philosophical intuition where the epistemicity of these new types of information stand with respect to the irrefutable and defeasible ones.

In clearer terms, we intend to discuss the basis for a system in which:

(a) conflicting situations can be tolerated, explicitly captured and reported (this we call *explicit paraconsistency*), and
(b) intuitively relevant knowledge can be obtained from this conflicting situation, from the rest of the global theory which is not directly affected by contradictory information and also from the contradictory information and parts of the theory directly and indirectly affected by contradictory information (this we call *reasoning beyond paraconsistency*).

This paper is organized as follows. In the next section we will introduce the inconsistency handling approaches. In Section 3 we will explore the syntax and semantics of the logic we propose. In the final section we conclude with a summarization and extension of our work.

2 Introducing the new elements of Approach $\mathcal{C} - \mathcal{C}_d$

The question which arises very often in the logic community is how to handle contradiction. The consensus of opinion in the logic community is that contradiction is undesirable. The approach has been to completely free databases of contradiction, and try to eradicate contradiction from databases by any means possible. Many of them seem to agree that contradiction should not exist in a database and must be resolved somehow [1, 15, 14]. But some researchers [2, 13, 5, 17] in this area contend the prevailing notion amongst many of the researchers on the basis of the rationale that inconsistency in large databases is inevitable. On the other hand, checking or guaranteeing consistency in large databases is very expensive if at all possible. So to be more pragmatic we must have some means to reason in the presence of inconsistency.

We propose a new approach to handle inconsistency. We start with the basic language of positive logic programs [4] and introduce 'explicit' (or 'strong') negation[1]. Earlier, positive logic programs have been extended by negation, but the semantics given to it is not that of explicit negation, but of *negation as failure* [3]. We introduce in the language of positive logic programs the notion of explicit negation, where only the existence of a negated fact in the knowledge-base will make the strong notion of negation manifest itself in the theory, instead of *closed world assumption* [16] which makes negation as failure manifest in the theory. (For more on the rationale behind introducing explicit negation to logic programs see [6, 12].)

[1] Explicit negation ($\sim$) is different from classical negation ($\neg$) (see [12]).

Existence of a thesis and its explicit negation, together in a theory, gives rise to contradiction. Because we have two kinds of negations we also have two kinds of inconsistency. The inconsistency arising on account of explicit negation is called *ontological inconsistency* and the inconsistency arising on account of negation by failure is called *epistemic inconsistency.* (Refer to [10, 8, 9, 7] for discussion on epistemic and ontological inconsistency.)

In this paper we will deal with ontological inconsistency by restricting our language to that of positive logic programs plus explicit negation[2]. To handle ontological inconsistency, we introduce a new connective $\mathcal{C}$ to the existing language of positive logic programs plus explicit negation (denoted as $\sim$). $\mathcal{C}\alpha$ means *α (resp. $\sim \alpha$) is inconsistent or α (resp. $\sim \alpha$) is in contradiction)*, i.e. there is contradictory information available about α (resp. $\sim \alpha$) in a theory. By our semantics if α and $\sim \alpha$ are in the same theory , we replace them by $\mathcal{C}\alpha$. α is an atom.

Unlike some of the other logics handling inconsistency (see [8, 17]), we do not allow contradictory information to prevail in the theory but still maintain paraconsistency. By an introspective method we check the theory for the presence of contradiction. If contradiction is present we replace it by a sentence depicting it. In this way we avoid all the repercussions[3] of having contradictions in a theory and at the same time maintain the privileges[4] of paraconsistency. We name this particular concept of handling inconsistency *explicit paraconsistency* and more formally **Approach $\mathcal{C}$**.

Approach $\mathcal{C}$ based systems are paraconsistent without the presence of explicit inconsistency. Considering a traditional paraconsistent system, if a set of formulae $\Sigma = \{\sim a, a\}$ and $\models_{para}$ denotes logical consequence in the system, then $\Sigma \models_{para} a$ (resp. $\sim a$), but $\Sigma \not\models_{para} b$ (the theory is not trivialized by concluding everything from inconsistency). *Approach $\mathcal{C}$* integrated paraconsistent systems behaves differently. If $\models_{\mathcal{C}}$ denotes the logical consequence for the system, then $\Sigma \not\models_{\mathcal{C}} a$ (resp. $\sim a$), $\Sigma \not\models_{\mathcal{C}} b$ and $\Sigma \models_{\mathcal{C}} \mathcal{C}a$. By this approach inconsistent pairs (such as a and $\sim a$) are not explicitly present in the theories but replaced by the new element $\mathcal{C}a$. We refer to formulas like $\mathcal{C}a$ as 'paraconsistent elements' as they contribute to the paraconsistency of the system. There are many advantages to this approach. This we will discuss in the nex section.

In most of the logics handling inconsistencies [2, 13, 17], none distinguishes between conclusions drawn from inconsistent and consistent information. Moreover there is no logical mechanism in these logics to allow or disallow conclusions from inconsistent information as the situation demands.

The philosophical question which we face here is whether we should at all distinguish between conclusions arrived from inconsistent information and con-

[2] We handle both ontological and epistemic inconsistency in the follow up paper [9] in the context of extended logic programs.

[3] Contradictory pairs of information explicitly present in a theory leads to inferences being made from them. This may propagate conclusions affected by contradictions without the knowledge of the reasoner. This is not warranted for.

[4] Ability to carry out nontrivial reasoning in presence of inconsistency.

clusions arrived from consistent information. Existing paraconsistent logics have not dealt with this fundamental question. We argue that we should distinguish between conclusions from inconsistent and consistent information as we understand that the conclusions derived from inconsistency have different epistemic strength than the ones derived from consistent information. (This argument is somewhat similar to [13] which distinguishes between irrefutable and defeasible conclusions arrived via irrefutable and defeasible rules respectively.) The logic which we propose here, deals with the issue of the distinction of conclusions drawn from inconsistent and consistent information. The logical system we propose moreover is a compact system where logical devices can be easily built in to give us the option to allow or disallow conclusions from inconsistent information as the situation arise. So in our proposed framework not only we go beyond inconsistency by introducing paraconsistent techniques, but we go beyond paraconsistency to give an extended logical framework which can handle *reasoning beyond paraconsistency.*

We introduce a new connective $\mathcal{C}_d$ for this purpose of reasoning beyond paraconsistency. A formula $\mathcal{C}_d\alpha$ means that *α is the consequence of a premise of a rule*[5] *affected by inconsistent information.* There are three possible ways in which the premise of a rule can be affected by contradictory information: (1) The premise of a rule can have a contradiction i.e., can have a formula of the form $\mathcal{C}\alpha$ in it and also $\mathcal{C}\alpha$ as a consequence of the rest of the theory containing the rule; (2) the premise can have a formula of the form $\mathcal{C}_d\alpha$ and a consequence $\mathcal{C}_d\alpha$ of the rest of the theory; or (3) the premise can have a formula α and the rest of the theory has $\mathcal{C}\alpha$ or $\mathcal{C}_d\alpha$ as a consequence. In allowing paraconsistent formulas like $\mathcal{C}\alpha$ and $\mathcal{C}_d\alpha$ we are also enhancing the expressivity of the language.

The formal strategy we develop to the proposed concept of *reasoning beyond paraconsistency* is termed as **Approach $\mathcal{C}_d$**. We name the combination of **Approach $\mathcal{C}$** and **Approach $\mathcal{C}_d$**: **Approach $\mathcal{C} - \mathcal{C}_d$**. Application of **Approach $\mathcal{C} - \mathcal{C}_d$** to the language of positive logic programs [4, 11] with explicit negation gives us a new language which we call *Paraconsistent Specifications (PSs).* We will formally define this in Section 3.

In [8] we have discussed the advantages and appropriateness of handling inconsistency by *Approach $\mathcal{C} - \mathcal{C}_d$* in the light of the existing approaches of (a) paraconsistency, (b) traditional information processing and (c) multi-valued logics. Due to space constraint we are unable to include it here.

3 Syntax & Semantics

In this section we will describe the language and define the model theoretic semantics.

Let us consider a language $\mathcal{L}$ consisting of atomic sentences a, b, c, ..., p, q, ..., logical connectives $\wedge$ (and), $\sim$ (explicit negation), the new ones $\mathcal{C}$, $\mathcal{C}_d$ and the

[5] An extended positive logic program rule, extended by explicit negation, the paraconsistent connective $\mathcal{C}$ and the connective handling reasoning beyond paraconsistency $\mathcal{C}_d$.

connective $\leftarrow$ which we call *inference implication*. Formulas of $\mathcal{L}$ will be defined in the following way. Formula of the form p will be called *atoms*. A *literal* is an atom α or a negated atom of the form $\sim \alpha$. We use Φ to denote the set of all literals. *Paraconsistent literals* are formulas of the form $\mathcal{C}\phi$ and $\mathcal{C}_d\phi$, where $\phi \in \Phi$. We use Ψ to denote the set of all paraconsistent literals.

We will restrict ourself to a subset of the language of $\mathcal{L}$, which we call *Paraconsistent Specifications* (let us denote it as PS). We allow contradictions to prevail in a theory of a PS by cautiously capturing it by $\mathcal{C}$ and capture the 'inferentially implicated' consequences (derivations) from premises having contradictions or derivations from premises affected by contradictions by $\mathcal{C}_d$. The formal notion of such a specification which is a syntactical extension of a *definite (positive) logic program* [11] is captured by the following definition:

Definition 1 : Paraconsistent Specification (PS). By a paraconsistent specification P we will mean a collection of clauses of the form

$$\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$$

where $\phi \in \Phi$, ψ_is are formulae belonging to the set of formulae $\Phi \cup \Psi$ of the language $\mathcal{L}$ and $m \geq 0$. □

Now we will propose a model theoretic semantics for paraconsistent specifications. The approach is somewhat different from the existing model theoretic semantics of logic programs, as we allow the theories to be paraconsistent i.e., allow for nontrivial reasoning in presence of contradiction. We define the semantics of our logic by an extension of the standard notion of *interpretations*, which we call *p-interpretations*. Here *'p'* stands for two notions: (1) PS-specific, i.e. because the existence of the elements in a p-interpretation depends on the PS (for paraconsistent specification) being considered and (2) paraconsistent, i.e. because in a p-interpretation we also allow paraconsistent literals along with literals. Here we adhere to a two-valued semantics.

In a knowledge-base α and $\sim \alpha$ can coexist. This situation arises, say, when two knowledge-bases are collapsed together, or two experts feeding inconsistent information to a knowledge-base. To handle this situation, the crucial facet of the stance we are taking in our formalism is that, literals α and $\sim \alpha$ are not related to each other in the usual classical logic sense, i.e. if α is *true* we should not conclude that $\sim \alpha$ is *false*. We take the stance that the truth-theoretic valuation ('truth' or 'falsity') of α and that of $\sim \alpha$ in an interpretation are to be independent of one another. The existential status of α is to be decoupled from that of $\sim \alpha$: the facticity of α is not to preclude that of $\sim \alpha$.

In our semantical definitions, we are introducing some revisions to the existing concepts of *truth-valuation in an interpretation* of a formula and *satisfaction by an interpretation* of a formula. They are originally the same in the traditional model-theoretic semantics of classical logic. But, we make a difference between them. The truth-value assignment to a propositional symbol is arbitrary. So in our semantics, as we have disassociated the assignment of truth values of the literals α and $\sim \alpha$, they can both get the value *true* or *false* simultaneously. By differentiating between the concept of *truth-valuation in an interpretation* and *satisfaction by an interpretation*, which we do by restricting the *satisfaction* by some conditions coupled with the truth-value assignment to *true*, we are able to make α and $\sim\alpha$ not *satisfied by an interpretation*, though letting them be simultaneously *true* in the same interpretation.

Our semantics have a two-tier approach. In the first tier, that of assignment of truth-values in an interpretation, only literals pertaining to the PS being considered are arbitrarily assigned the truth-values *true* or *false*. In the second tier, satisfaction of literals by an interpretation are guided by certain conditions. Thus all literals which are *true* in an interpretation may not be satisfied by an interpretation. In the second tier, in addition to the literals already satisfied by an interpretation, paraconsistent literals are brought into the picture. We now have paraconsistent literals satisfied by an interpretation depending upon the PS and the corresponding interpretation being considered. Hence we see that satisfaction by an interpretation is guided by the specific PS for which the interpretation is being considered. Thus we rename *satisfaction by an interpretation* as *satisfaction by an interpretation w.r.t. a PS.* Finally, we define a *p-interpretation of a PS P* as a set which contains all the literals and paraconsistent literals satisfied by the satisfiability relations (i.e. satisfiability conditions) of *satisfaction by an interpretation w.r.t. the PS P.* Models are based on these p-interpretations. This would be better understood with the formalization that follows.

Paraconsistent literals of the form $\mathcal{C}_d\phi$'s existence in a p-interpretation is guided by the rules of a PS. This is clear from the meaning of $\mathcal{C}_d\phi$.

It is crucial to note, however, that while we may well have it that α and $\sim\alpha$ both can be assigned the truth-value *true* by an interpretation, we shall certainly never have it that $\alpha\wedge\sim\alpha$ is satisfied in an interpretation. Our perspective is that two mutually inconsistent states of affairs might well both be realized, whereas a single *self-inconsistent* state of affairs can never be realized. Contradictions can be realized distributively but not collectively: *self-contradiction* must be excluded. We shall always have $\alpha\wedge\sim\alpha$ unsatisfied by an interpretation. This will have its affect on the application of the logical connective $\wedge$ over formulas.

For the cases where α (resp. $\sim\alpha$) exist by itself in a knowledge base, the truth-value *true* is assigned to α (resp. $\sim\alpha$) in an interpretation. But the satisfaction of α (resp. $\sim\alpha$) by an interpretation will be determined by first checking if $\mathcal{C}\alpha$ is not satisfied by the interpretation. Thus we define the satisfaction of α (resp. $\sim\alpha$) by an interpretation with respect to the satisfaction of $\mathcal{C}\alpha$.

In the case, where α (resp. $\sim\alpha$) is satisfied by an interpretation, the relation between α and $\sim\alpha$ is in the usual sense of classical logic. $\sim\alpha$ (resp. α) cannot be satisfied by an interpretation when α (resp. $\sim\alpha$) is satisfied by the interpretation.

We will now inductively define the formal notion of *truth-valuation in an interpretation* and *satisfaction by an interpretation w.r.t. a PS.* The definition of truth-valuation in an interpretation gives the truth-valuation of literals of our language restricted to the PS P being considered. Based on this the definition of satisfaction by an interpretation w.r.t. the PS P, of literals and paraconsistent literals is given. Based on this we define a p-interpretation of the PS P and then model of a PS P.

An interpretation I for a PS P is a *Herbrand-like* interpretation. It can only contain elements of the set Φ_p , which consist of all the literals of our language $\mathcal{L}$, i.e. Φ, restricted to the PS P. For example, the set $\Phi_p = \{a, b, \sim b\}$ for the PS $P = \{a \leftarrow b;\ \sim b \leftarrow a\}$. We define a *p-interpretation* I_p inductively on the basis of the following two definitions: truth-valuation in an interpretation and satisfaction by an interpretation w.r.t. a PS P.

Definition 2 : Truth-valuation in an interpretation.
Truth-valuation in an interpretation is defined as follows:

> Any literal ϕ or its complement $\tilde{\phi}$[6] has a truth-value **true** (resp. **false**) in an interpretation independent of each other. □

Remark. Our interpretations will only contain the elements from the set Φ_p which are **true**. Any other elements in Φ_p which are not in the interpretation are **false**.

Definition 3 : Satisfaction by an interpretation w.r.t. a PS P.
Let $\phi, \phi_c \in \Phi$, $\psi \in \Phi \cup \Psi$ and $\tilde{\phi}$ be the complement of ϕ (i.e. if ϕ is α, a positive literal (an atom), $\tilde{\phi}$ is $\sim \alpha$ or vice versa). We write $I \models_{C-C_d} \psi$ to say that a formula ψ is *satisfied* by the interpretation I w.r.t. a PS P and $I \not\models_{C-C_d} \psi$ to say that a formula ψ is *not satisfied* by the interpretation I w.r.t. a PS P.

(1) $I \models_{C-C_d} C\phi$ if ϕ =**true** and $\tilde{\phi}$ =**true** in the interpretation I.
(2) $I \models_{C-C_d} \phi$ (resp. $\tilde{\phi}$) if $I \not\models_{C-C_d} C\phi$, $I \not\models_{C-C_d} C_d\phi$ (resp. $C_d\tilde{\phi}$) and ϕ (resp. $\tilde{\phi}$ =) **true** in the interpretation I.
(3) $I \models_{C-C_d} C_d\phi$ if
there exists a formula $\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ in the PS P, such that
  (i) $I \models_{C-C_d} \psi_i$ for all $i = 1, \ldots, m$ and
  (ii) there exists a formula $C\phi_c$ (or $C_d\phi_c$) $\in \{\psi_1, \ldots, \psi_m\}$
or
  (i) $I \models_{C-C_d} C\phi_c$ (resp. $C_d\phi_c$), such that, $\phi_c \in \{\psi_1, \ldots, \psi_m\}$ and
  (ii) $I \models_{C-C_d} \psi_i$ for the rest of the ψ_is.
(4) $I \models_{C-C_d} \phi$ if
there exists a formula $\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ in the PS P, such that
  (i) ϕ =**true** in the interpretation I
  (ii) $I \models_{C-C_d} \psi_i$ for all $i = 1, \ldots, m$
  (iii) $I \not\models_{C-C_d} C\phi$
  (iv) there does not exist any formula of the form $C\phi_c$ or $C_d\phi_c \in \{\psi_1, \ldots, \psi_m\}$ and
  (v) $I \not\models_{C-C_d} C\phi_c$ *(resp.* $C_d\phi_c$*)*, when $\phi_c \in \{\psi_1, \ldots, \psi_m\}$.
(5) these are the only ways by which a formula ϕ can be satisfied by an interpretation I w.r.t. a PS P. □

[6] $\phi \in \Phi$, (Φ is the set of all literals in $\mathcal{L}$, but here it is restricted to Φ_p). ϕ is of the form α or $\sim \alpha$, where α is an atom. If $\phi = \alpha$ then $\tilde{\phi} =\sim \alpha$ and vice versa.

Remark. In this work as in [2, 6], we give a non-classical interpretation to the '$\leftarrow$' symbol, i.e., we do not treat it as a classical material implication. In particular, the equivalence of $\alpha_1 \vee \sim \alpha_2$ and $\alpha_1 \leftarrow \alpha_2$ does not hold. This is necessary because, in general, $\alpha_1 \vee \sim \alpha_1$ may not be a tautology. This formula asserts (in an intuitionistic fashion) the sentence: *It is* **known** *that α_1 is true OR it is* **known** *that $\sim \alpha_1$ is true.*

Definition 4 : P-interpretation.
A p-interpretation I_p is a set of literals and paraconsistent literals satisfied by an interpretation I with respect to a PS P. □

Proposition 5. *Any formula ϕ ($\in \Phi_p$) 'true in an interpretation of a PS P' is not necessarily 'satisfied by the interpretation w.r.t. P'.* □

Definition 6 : Satisfaction by a p-interpretation of a rule in a PS.
A rule R in a PS P of the form:

$$\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$$

is satisfied by a p-interpretation I_p iff

$$\psi_i \notin I_p,\, \mathcal{C}\psi_i \notin I_p \text{ and } \mathcal{C}_d\psi_i \notin I_p \text{ for some } 1 \leq i \leq m$$
$$\text{or}$$
$$\phi \in I_p \text{ or } \mathcal{C}\phi \in I_p \text{ or } \mathcal{C}_d\phi \in I_p. \;\square$$

Definition 7 : Model of a PS.
We say that a p-interpretation I_p, is a 'model' M, of a PS P iff every rule R in P is satisfied by I_p. □

Definition 8 : Minimal model of a PS.
A model M of a PS P is 'minimal', if no proper subset of M is a model of P. A minimal model (denoted by M_p) gives the intended semantics of a PS P. □

Definition 9 : Entailment of a formula from a PS.
A PS P 'entails' a formula F, denoted by $P \models_{\mathcal{C}-\mathcal{C}_d} F$, iff every model of P is also a model of F. □

Theorem 10. *Every PS P has a unique minimal model M_p.*

Example 1. Consider the PS P given below:

$$\{a \leftarrow b;\ \sim b \leftarrow a\}$$

Let $I_1 = \emptyset$ be an interpretation. The corresponding p-interpretation $I_{p1} = \emptyset$. I_{p1} is a model of P.
Let $I_2 = \{a\}$ be an interpretation. The corresponding p-interpretation $I_{p2} = \{a\}$. I_{p2} is not a model of P.
Let $I_3 = \{a, b\}$ be an interpretation. The corresponding p-interpretation $I_{p3} = \{a, b\}$. I_{p3} is not a model of P.
Let $I_4 = \{a, b, \sim b\}$ be an interpretation. The corresponding p-interpretation $I_{p4} = \{\mathcal{C}b, \mathcal{C}_d a, \mathcal{C}_d \neg b\}$. I_{p4} is a model of P.
Thus the minimal model of P: $M_p = \emptyset$. □

Example 2. Consider the PS P given below:

$$\{p \leftarrow; \sim p \leftarrow\}$$

Let $I_1 = \emptyset$ be an interpretation. The corresponding p-interpretation $I_{p1} = \emptyset$. I_{p1} is not a model of P.
Let $I_2 = \{p\}$ be an interpretation. The corresponding p-interpretation $I_{p2} = \{p\}$. I_{p2} is not a model of P.
Let $I_3 = \{p, \sim p\}$ be an interpretation. The corresponding p-interpretation $I_{p3} = \{\mathcal{C}p\}$. I_{p3} is a model of P.
Thus the minimal model of P: $M_p = \{\mathcal{C}p\}$. □

Example 3. Consider the PS P given below:

$$\{p \leftarrow; \; q \leftarrow; \; r \leftarrow q; \; \sim q \leftarrow r\}$$

Let $I = \{p, q, r, \sim q\}$ be an interpretation. The corresponding p-interpretation $I_p = \{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q, p\}$. I_p is a model of P.
There are no other models for P. Thus the minimal model of P: $M_p = \{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q, p\}$. □

4 Conclusion

In this paper we have introduced two inconsistency handling strategies, *Approach* $\mathcal{C}$ and *Approach* $\mathcal{C}_d$. We applied the strategies to the language of positive logic programs with explicit negation. This enhanced the language, enabling the expanded language PS to handle inconsistency in a pragmatic and satisfactory way. We have also defined a fixpoint semantics for PS (in the full paper [8]), which we could not present here because of space-constraint.

This work can be further expanded by applying the inconsistency handling strategies to nonmonotonic frameworks, where the theories collapse in presence of contradictions. In [9] we have applied the inconsistency handling strategies to the nonmonotonic framework of 'extended logic programs' [6].

References

1. Carlos E. Alchourron, Peter Gardenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Symbolic Logic*, 50(2):510–530, June, 1985.
2. H. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computing Science*, 68:135–154, 1989.
3. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logics and Data Bases*, pages 293–322. Plenum Press, New York, 1978.
4. M. Van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 23(4):37—54, 1976.
5. Dov Gabbay and Anthony Hunter. Making inconsistency respectable: A logical framework for inconsistency in reasoning, part i - a position paper. In *Lecture Notes in Artificial Intelligence (535), Proceedings of the International Workshop FAIR-91*, pages 19–32. Springer-Verlag, 1991.

6. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the 7th. International Conference on Logic Programming*, pages 579–597. MIT Press, 1990.
7. Suryanil Ghosh. Applying Approach $\mathcal{C} - \mathcal{C}_d$ to Extended Logic Programs. manuscript, 1994. Dept. of Computing Science, University of Alberta.
8. Suryanil Ghosh. Approach $\mathcal{C} - \mathcal{C}_d$: A new contradiction handling strategy. manuscript, 1994. Dept. of Computing Science, University of Alberta, Canada.
9. Suryanil Ghosh. Approach $\mathcal{C} - \mathcal{C}_d$: A new contradiction handling strategy & Extended Logic Programs. In *Proceedings of the Third Golden West International Conference on Intelligent Systems.* Kluwer Academic Press, June 6-8 1994.
10. Suryanil Ghosh. Paraconsistency and beyond: Issues and approaches in reasoning. manuscript of Ph.D. thesis (in preparation), 1994. Dept. of Computing Science, University of Alberta, Canada.
11. J. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.
12. Jack Minker and Carolina Ruiz. On extended disjunctive logic programs. In J. Komorowski and Z. W. Ras, editors, *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems*, pages 1–18. Springer-Verlag, 1993. Lecture notes in AI, June 1993.
13. Tarcisio Pequeno and Arthur Buchsbaum. The logic of epistemic inconsistency. In *Proc. of Second Intl. Conference of Principles of Knowledge Representation and Reasoning*, pages 453–460. Morgan Kaufmann, 1991.
14. L. M. Pereira, J. J. Alferes, and J. N. Aparicio. Contradiction removal within well founded semantics. In *Proceedings, 1st. Intl. Workshop on Logic Programming and Nonmonotonic reasoning.* MIT Press, July, 1991.
15. S. G. Pimentel and W. L. Rodi. Belief revision and paraconsistency in a logic programming framework. In *Proceedings, 1st. Intl. Workshop on Logic Programming and Nonmonotonic reasoning.* MIT Press, July, 1991.
16. R. Reiter. On closed-world databases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.
17. Gerd Wagner. Reasoning with inconsistency in extended deductive databases. In *Proc. of the 2nd. Intl. Workshop of Logic Programming and Nonmonotonic Reasoning, Lisbon, Portugal*, pages 300–315. MIT Press, 1993.

BDDs and Automated Deduction

Jean Goubault[1] and Joachim Posegga[2]

[1] Bull Corporate Research Center, rue Jean Jaurès, 78340 Les Clayes-sous-Bois, France, (Jean.Goubault@frcl.bull.fr)

[2] Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, 76128 Karlsruhe, Germany, (posegga@ira.uka.de)

Abstract. BDDs (binary decision diagrams) are a very successful tool for handling boolean functions, but one which has not yet attracted the attention of many automated deduction specialists. We give an overview of BDDs from an automated deduction perspective, showing what can be done with them in propositional and first-order logic, and discuss the parallels to well-known methods like tableaux and resolution.

1 Introduction

BDDs (binary decision diagrams) are a very successful tool for handling boolean functions, which has been used extensively in hardware verification (7), truth-maintenance systems (18) and various other domains; often they have superseded previously-known methods. Surprisingly, they have not yet attracted the attention of many automated deduction specialists. Our aim in this paper is to show that BDDs can be profitably used in this domain, too.

Although we present briefly some BDD-based proof methods, we don't emphasize any particular technique. We give an overview of BDDs from an automated deduction perspective, and demonstrate various ways in which they can be used in propositional and first-order classical logic. As regards the propositional part, we recall what can be done with BDDs and how, referring the reader to the literature for complementary results and facts. The first-order part is mostly new, and aims at showing that BDDs are a rich structure that can be used in several different ways, including tableaux-like and resolution-like techniques.

The plan of the paper is as follows: Section 2 defines BDDs, recalls their history briefly, and goes on defining logical operations and reductions on BDDs. Section 3 discusses their theoretical and practical time and space complexities. In Section 4, we show that BDDs relate to semantic tableaux, so that both approaches can benefit from each other. We then present some principles for proving with BDDs in first-order logic in Section 5; the first one is similar to tableaux, the last one has some common points with resolution, and each corresponds to a slight change of point of view on BDDs. Section 6 is the conclusion.

2 What are BDDs?

BDDs are nested *if-then-else*-expressions represented as graphs, which have been very successfully applied to various domains, e.g. hardware verification (see for instance (7; 19)). These *if-then-else*-expressions are basically a case analysis over the truth value of atoms in a formula. This is a very natural way of reasoning about logical formulæ, as it resembles the way humans often deal with logic: one fixes a certain proposition and reasons about the consequences that follow if it is true, and if it is false. BDDs provide a convenient and efficient tool for implementing this form of reasoning.

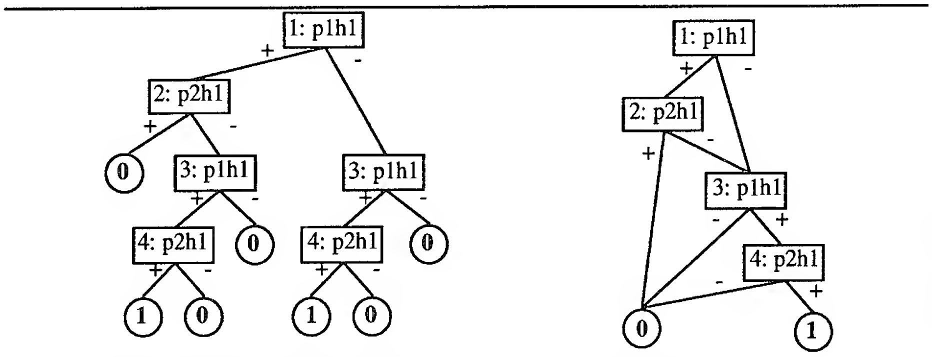


Figure 1. An *If-then-else*-expression and its representation as a BDD.

The literature usually refers to an early paper by Claude Shannon (24) as the origin of BDDs. Shannon introduced a special *if–then–else* normal form for boolean functions. The objective was to have a convenient and efficient representation for implementing boolean functions as switching circuits. The idea of using *if–then–else* expressions was not new: the principle of Shannon's co-factoring was already described in 1854 by Boole (3). *if-then-else* normal forms were also considered by Church (9, §24, pp. 129ff), who introduced a ternary operator of the form $[A, B, C]$ and called it a *"conditioned disjunction"*. Church's interest in this connective is based on the observation that it provides a primitive basis of propositional logic.

It is hard to command a view on the enormous amount of literature on the subject; Bryant (6) gives a good introduction.

Various notations for expressing "*if A then B else C*" are used in the literature; we will use here a notation that resembles Prolog's syntax:

Definition 1 *if-then-else*-**connective.** $(A \rightarrow B; C) \overset{\text{def}}{=} (A{\rightarrow}B)\wedge(\neg A{\rightarrow}C)$

In the *if-then-else*-expressions BDDs represent, no other logical connectives beside *if-then-else* are allowed. To ease exposition, we will not distinguish between a BDD and the *if-then-else*-expression it represents, but treat BDDs as ordinary logical formulæ. Recall that propositional atoms are propositional variables, and first-order atoms are applications of predicate symbols to lists of terms.

Definition 2 BDDs. Let $\mathcal{L}$ be the language of (propositional/first-order) logic and $\mathcal{L}_{At}$ the atomic formulæ of $\mathcal{L}$. Assume further, that the language $\mathcal{L}$ does not contain the atomic truth values "**1**" (*true*) and "**0**" (*false*).
The set of **BDDs** is denoted by $\mathcal{BDD}$ and defined as the smallest set such that
a) $\mathbf{1}, \mathbf{0} \in \mathcal{BDD}$, and b) *if* $\mathcal{A}, \mathcal{B} \in \mathcal{BDD}$ *and* $\phi \in \mathcal{L}_{At}$ *then* $(\phi \rightarrow \mathcal{A}; \mathcal{B}) \in \mathcal{BDD}$.
We shall use letters of the calligraphic alphabet $(\mathcal{A}, \mathcal{B}, \ldots)$ to denote the elements of $\mathcal{BDD}$ in the sequel.

The formulæ defined above can be conveniently represented as binary trees, or binary graphs: each condition in the *if-then-else*–expression is drawn as a node, and two edges, the *positive edge* (labeled with "+") and the *negative edge* (labeled with "–"), lead to the "then"-part, and the "else"-part, respectively.

Fig. 1 shows an example: a binary tree that corresponds to

$$\begin{aligned}(p1h1 \rightarrow (p2h1 &\rightarrow \mathbf{0}\\ &;(p1h1 \rightarrow (p2h1 \rightarrow \mathbf{1};\mathbf{0});\mathbf{0}))\\ ;(p1h1 &\rightarrow (p2h1 \rightarrow \mathbf{1};\mathbf{0});\mathbf{0}))\end{aligned}$$

is on the left hand side. This formula is logically equivalent to the smallest instance

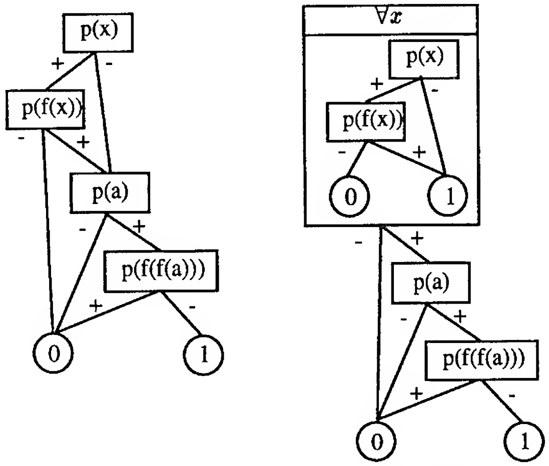


Figure 2. BDDs for $(\forall(x)p(x) \to p(f(x)))\wedge p(a)\wedge\neg p(f(f(a)))$

of the pigeon-hole formulæ $((p1h1\to\neg p2h1)\wedge p1h1\wedge p2h1)$. The nodes are labeled by $(1,2,\ldots)$ to identify them in the sequel. This formula can also be represented in a more compact way if a graph instead of a tree is used: such a graph shares identical subtrees, and is therefore much smaller. This is shown on the right hand side.The leaves (which are, by definition, either "**1**" or "**0**") are also shared.

Note, that both the tree and the graph in Fig. 1 denote the same *if-then-else*–expression; they are just different representations of it. In the sequel, we will assume BDDs are always represented as graphs that share maximally, i.e., those graphs do not contain multiple occurrences of the same subgraph. Note that the size of a graph and its corresponding tree can differ exponentially; it is in practice therefore very important to use a graph representation: graphical representation of the formulæ for pigeon-hole(7), for instance, a has about 400 nodes, but a tree 10^{34} (20).

BDDs form a logical basis for propositional logic (see, e.g., Church (9)). We will say that a BDD *represents a formula*, if it is logically equivalent to it. BDDs can be used to represent also quantifier-free formulæ of first-order logic (see the BDD on the left-hand side of Fig. 2; the right-hand side will be discussed in Section 5.1):

Lemma 3. *For each quantifier-free formula F of first-order logic, there is a logically equivalent BDD $\mathcal{F}$.*

Paths in BDDs are an important concept that relates a BDD to its semantics; A *path* in a BDD is a sequence of signed nodes, that leads from the root to one of the leaves. Paths that lead to **1**-leaves are called **1**-paths, and paths that lead to **0**-leaves are called **0**-paths.

A path can be regarded as a conjunction of literals: the $-$-signs are treated as negation, and $+$-signs are dropped. A path is in this case either *consistent*, or *inconsistent*. Example: the right-most **1**-path "¬p1h1 ∧ p1h1 ∧ p2h1" of the BDD in Fig. 1 is inconsistent.

Proposition 4. *Let $\mathcal{G} \in \mathcal{BDD}$; then*
(1) $\mathcal{G}$ is logically equivalent to the disjunction of its **1***-paths, and*
(2) $\neg\mathcal{G}$ is logically equivalent to the disjunction of its **0***-paths.*

From this proposition (see (20) for a proof) follows that a BDD "contains" both a disjunctive normal form for itself (regarded as a formula), as well as a DNF for its negation. If we regard **0**-paths as disjunctions of literals with the signs reversed instead, we can also have a CNF: the conjunction of all **0**-paths. The **1**-paths form, from this point of view, a CNF for the negated BDD.

Let us take the DNF perspective; a procedure for determining the satisfiability and falsifiability of a BDD is easily set up:

Corollary 5. *Let $\mathcal{G} \in \mathcal{BDD}$;*
(1) If there is a substitution σ, such that all $\mathbf{1}$-paths of $\mathcal{G}\sigma$ are inconsistent, then $\mathcal{G}$ is inconsistent.
(2) Conversely, if there is a substitution σ, such that all $\mathbf{0}$-paths of $\mathcal{G}\sigma$ are inconsistent, then $\mathcal{G}$ is a tautology.

Example: the BDD in Fig. 1 is inconsistent, as all its $\mathbf{1}$-paths are inconsistent.

There are several ways to convert an arbitrary logical formula into a logically equivalent BDD; one of them is to recursively rewrite the subformulæ into BDDs and combine those according to the formulæ's connectives. The following properties of the *if-then-else*–operator show how this can be done (20):

Proposition 6.
(1) $\neg(P \rightarrow \mathcal{Q};\mathcal{R}) \Leftrightarrow (P \rightarrow \neg\mathcal{Q};\neg\mathcal{R})$
(2) If $(P \rightarrow \mathcal{Q};\mathcal{R}) \in \mathcal{BDD}$, $F \in (\mathcal{L} \cup \mathcal{BDD})$ and "$\circ$" is any binary logical connective, then: $(P \rightarrow \mathcal{Q};\mathcal{R}) \circ F \Leftrightarrow (P \rightarrow (\mathcal{Q} \circ F);(\mathcal{R} \circ F))$

(1) shows how to move negation towards the leaves of BDDs; those negation signs can be removed at the leaves of a BDD, since those are always either $\mathbf{0}$ or $\mathbf{1}$. Thus, a BDD can be negated by inverting its leaves, an operation that can be carried out without descending to the leaves: if an appropriate datastructure is set up, this can be implemented to require constant effort only. (2) is a rather powerful rule that applies to every binary connective. To compute a conjunction of two BDDs $\mathcal{A}$ and $\mathcal{B}$, for example, we recursively apply rule 2 to $\mathcal{A}\wedge\mathcal{B}$ until we have moved "$\ldots \wedge\mathcal{B}$" to the leaves of $\mathcal{A}$. There it can be simplified since $\mathbf{1}\wedge\mathcal{B}$ is $\mathcal{B}$ and $\mathbf{0}\wedge\mathcal{B}$ is $\mathbf{0}$. This means that conjunctively combining two BDDs corresponds to inserting one BDD for the $\mathbf{1}$-leaf of the other. Symmetrically, disjunctions are handled by replacing the $\mathbf{0}$-leaf. Both operations can be carried out in constant time.

2.1 Orderings in BDDs

BDDs, as defined above, may still contain much redundancy: the paths may have multiple occurrences of atoms, and there may be multiple occurrences of subgraphs representing propositionally equivalent subformulas.

Definition 7. A BDD $\mathcal{G}$ is called *reduced* if no atom occurs more than once in a path of $\mathcal{G}$, and if $\mathcal{G}$ does not contain multiple occurrences of one of its subgraphs.

E.g., the left graph in Fig. 1 violates both conditions, the right graph only the second one.

It is of course possible to eliminate these redundancies in a given graph, but it is more reasonable to avoid them in advance while constructing a BDD. The use of *ordered* BDDs is a well-known technique to achieve this:

Definition 8. Ordered BDDs / Reduced Ordered BDDs. Let $\mathcal{L}_{At}$ be the atoms in $\mathcal{L}$ and "<" be a total ordering on $\mathcal{L}_{At}$. A BDD $\mathcal{G}$ is called an *ordered BDD* if all its paths respect "<". An ordered BDD which is reduced is called a *reduced, ordered BDD (ROBDD).*

ROBDDs have the important property that all of their paths are consistent. This, together with a fixed ordering on atoms, gives a unique normal form for boolean functions. Note that, as a consequence, the derivation of a ROBDD for a given propositional formula is *NP*-hard, since every inconsistent formula will yield a ROBDD consisting of the single node "$\mathbf{0}$". In other words: the derivation of a ROBDD for a given formula is a decision procedure for propositional logic.

The derivation of ROBDDs is described extensively in the literature (4), so we only give a very brief account of it. Negation obeys the same equation as before, whereas logical connectives "$\circ$" obey new equations. Let $\mathcal{G}$ be the ROBDD $(A \rightarrow \mathcal{B};\mathcal{C})$, and $\mathcal{G}'$ be the ROBDD $(A' \rightarrow \mathcal{B}';\mathcal{C}')$, then:

- if $A < A'$, then $\mathcal{G} \circ \mathcal{G}' = \text{construct}(A, \mathcal{B} \circ \mathcal{G}', \mathcal{C} \circ \mathcal{G}')$;
- if $A > A'$, then $\mathcal{G} \circ \mathcal{G}' = \text{construct}(A', \mathcal{G} \circ \mathcal{B}', \mathcal{G} \circ \mathcal{C}')$;
- if $A = A'$, then $\mathcal{G} \circ \mathcal{G}' = \text{construct}(A, \mathcal{B} \circ \mathcal{B}', \mathcal{C} \circ \mathcal{C}')$,

where construct$(A, \mathcal{B}, \mathcal{C})$ is defined as $\mathcal{B}$ if $\mathcal{B}$ is identical to $\mathcal{C}$, and as $(A \to \mathcal{B}; \mathcal{C})$ otherwise. Reading these equations as rules from left to right, this recursively defines a procedure for computing $\mathcal{G} \circ \mathcal{G}'$, where the construct operation is responsible for maintaining the BDDs reduced.

By induction on the structure of quantifier-free formulæ, this leads to a procedure $\text{bdd}_< : \mathcal{L} \to \mathcal{BDD}$ that maps quantifier-free formulæ to ROBDDs[3].

3 Complexity

Building a non-ordered BDD is fast: we just combine BDDs for subformulæ by conjunction, disjunction or negation, which is done by constant-time operations on leaves. However, building ROBDDs, or reducing BDDs to a canonical form is more complex. We recall some results here.

First, operations on ROBDDs take time that is linear (negation), or quadratic (binary operations) in the size of the operands. Better representations even yield constant-time negation. But as BDDs form a logical basis, we can solve SAT, the propositional satisfiability problem (resp. coSAT, the propositional validity problem) by building a ROBDD and comparing it to **0** (resp. **1**). Thus, building a ROBDD is both NP-hard and coNP-hard.

Also, ROBDDs are exponentially-sized in the worst case, and so may need exponential time to be built. The exact bound is $O(2^n/n)$ for a ROBDD on n atomic formulæ. Unfortunately, this upper bound is also the mean value of the size of ROBDDs (10) if all are considered equally probable. In short, most ROBDDs are of exponential size.

The good surprise is that these huge ROBDDs very rarely occur in practice (10; 19). Moreover, ROBDDs have been in use in hardware verification for some time now, where the problems at hand can be NP-complete (model-checking CTL formulæ instance) or harder: model-checking in CTL* or in the modal μ-calculus (12) are PSPACE-complete. (PSPACE, the class of polynomial-space solvable problems, includes NP and coNP, but also the full polynomial hierarchy (14).) So, is there a trick that makes ROBDDs usable?

There is indeed one: the choice of a good ordering for the ROBDD. In most cases (19), there is a good ordering, i.e. one that not only produces an ROBDD of reasonable size, but also that produces the intermediate ROBDDs needed to build the final one in a reasonable amount of space, too. There are some problems for which no good ordering exists (the formula describing a binary multiplier, for instance (5)), and finding an optimal ordering is hard (the best known procedure needs exponential time (19)), but usually good heuristics, related to the application domain, have been developed.

In particular, when translating textual formulæ to ROBDDs, one such good ordering is based on the textual occurrences of propositional variables: if a variable A occurs left of a variable B, then let A be less than B. This attempts to mimick the layout of the textual formula inside the BDD so as to control the possible explosion of the ROBDD while it is built. Put it another way, this makes a ROBDD close enough to the non-ordered BDD built by replacement of leaves as in the previous section.

4 BDDs and Semantic Tableaux

It is widely unknown that non-ordered BDDs are closely related to tableaux-based proof procedures. We shortly investigate the relation between BDDs and semantic tableaux. We restrict ourselves to propositional logic in negation normal form, but the results can be carried forward to general, first-order formulæ.

[3] Most algorithms for deriving ROBDDs work for propositional formulæ only. For using them on quantifier-free formulæ of first-order logic, we regard atoms simply as propositional variables.

The key to considering BDDs and tableaux from a common perspective is to understand the relation between paths and branches: both have the purpose to represent potential models that are subsequently eliminated during the proof search. Assume we want to prove that a propositional formula F is inconsistent. We have already seen that the paths in a BDD are a disjunctive normal form for the BDD (seen as a logical formula), so if we have built a BDD for F, the disjunction of its paths is a disjunctive normal form for F. This is similar to semantic tableaux: the expansion of a tableau for a formula F can also be regarded as subsequently deriving a DNF for F.

Let for the rest of this section denote T_F a fully expanded tableau for a propositional formula F, and $\mathcal{F}$ denote a non-ordered BDD for F. $\#\,(T_F)$ denotes the set of all branches in T_F and $\Pi^{\mathbf{1}}_{\mathcal{F}}$ and $\Pi^{\mathbf{0}}_{\mathcal{F}}$ denote the set of all paths to $\mathbf{1}$–leaves and to $\mathbf{0}$–leaves in $\mathcal{F}$. The following notation will be convenient:

Definition 9. Let $k = k_1 \wedge \ldots \wedge k_n$ and $l = l_1 \wedge \ldots \wedge l_m$ be paths in a BDD or branches in a tableau, then $k \odot l \overset{\text{def}}{=} [k_1, \ldots, k_n, l_1, \ldots, l_m]$. "$\odot$" is carried forward to sets of paths or branches K and L, by defining: $K \odot L \overset{\text{def}}{=} \{k \odot l \mid k \in K \text{ and } l \in L\}$

Observe that $\{k\} \odot K$ means conjunctively adding k to K. A comparison of the treatment of conjunctions and disjunctions will show that branches and paths actually correspond: $\#\,(T_F)$ and $\Pi^{\mathbf{1}}_{\mathcal{F}}$ are logically equivalent when seen as formulæ. This can be proven formally by structural induction; we restrict the consideration to giving the key idea for the different cases in the recursion step of such a proof (see (21) for details).

4.1 Treatment of Conjunctions

Assume we have a formula of the form $A \wedge B$ and two fully expanded tableaux T_A and T_B; we can then build $T_{(A \wedge B)}$ by appending T_B at the end of each branch of T_A. This means: $\#\left(T_{(A \wedge B)}\right) = \#\,(T_A) \odot \#\,(T_B)$. In BDDs, a similar operation can be applied to represent a conjunction: if $\mathcal{A}$ and $\mathcal{B}$ are BDDs for A and B, then replacing the $\mathbf{1}$-leaf of $\mathcal{A}$ by $\mathcal{B}$ results in a conjunction of both BDDs. In terms of paths this replacement means: $\Pi^{\mathbf{1}}_{\mathcal{A}} \odot \Pi^{\mathbf{1}}_{\mathcal{B}}$ and corresponds to what happens in a tableau.

4.2 Treatment of Disjunctions

The treatment of disjunctions in BDDs is different from standard tableaux and corresponds to tableaux with *lemma generation* (see (11) on lemma generation in tableaux). One way to achieve this is to modify the standard β–rule of a tableau calculus to

$$\frac{A \vee B}{A \,|\, (\neg A) \wedge B}$$

With this rule, the branches of a fully expanded tableau $T_{A \vee B}$ will be $\#\,(T_A) \cup (\#\,(T_{\neg A}) \odot \#\,(T_B))$. For BDDs, we can handle disjunctions by replacing $\mathbf{0}$-leaves: a BDD for $A \vee B$ results from replacing the $\mathbf{0}$-leaf in $\mathcal{A}$ by $\mathcal{B}$. The resulting $\mathbf{1}$-paths are: $\Pi^{\mathbf{1}}_{\mathcal{A}} \cup (\Pi^{\mathbf{0}}_{\mathcal{A}} \odot \Pi^{\mathbf{1}}_{\mathcal{B}})$. As $\Pi^{\mathbf{0}}_{\mathcal{A}} = \Pi^{\mathbf{1}}_{\neg \mathcal{A}}$, we get the same result as in tableaux.

The above consideration shows an important difference in the representation of formulæ between BDDs and tableaux: in case of the above β–rule with lemmata, a tableau must expand the formula A and $\neg A$ separately and independently of each other. BDDs represent models and counter-models within the same structure; put simply, the replacement of $\mathbf{0}$–leaves for representing a disjunction $A \vee B$ with BDDs attaches the countermodels of $\mathcal{A}$ to $\mathcal{B}$. As models and counter models can be represented as efficient as models alone in tableaux, some effort of proof search with tableaux can be avoided with BDDs.

Remark. In some cases (especially in the first-order case) it might be desirable to *avoid* lemma generation; a simple "trick" within the BDD formalism turns lemma generation off: if $\mathcal{A}$ and $\mathcal{B}$ are two BDDs, then $(L \rightarrow \mathcal{A}; \mathcal{B})$, where L is a new atom that does not appear in either (A) or (B), can also be used for representing a disjunction. Note that this treatment of disjunctions preserves satisfiability, but not equivalence. If we treat every disjunction in this way, BDDs actually simulate semantic tableaux.

5 First-order BDDs

We shall see in this section how various proof search methods in first-order logic can be rebuilt with BDDs, and how we can improve on them.

To use BDDs in a first-order logic framework, we apply Herbrand's theorem on formulæ in Skolem normal form:

Proposition 10. *A formula $\forall x_1, \ldots, x_n \cdot M$, where M is quantifier-free, is unsatisfiable iff there is an integer k, and k substitutions $\sigma_1, \ldots, \sigma_k$, such that $M\sigma_1 \wedge \ldots \wedge M\sigma_k$ is propositionally unsatisfiable.*

If ρ_i denotes a generic renaming, mapping variables x to indexed variables x_i such that $x \neq y \vee i \neq j \rightarrow x_i \neq y_j$, we define the *k-fold copy* M^k of M as $M\rho_1 \wedge \ldots \wedge M\rho_k$.

So, $\forall x_1, \ldots, x_n \cdot M$ is unsatisfiable iff some instance of some k-fold copy of M is propositionally unsatisfiable; we can check this by reducing an ordered BDD for the instance of M^k, and comparing the result to **0**. As regards k, we cannot know it in advance, since first-order logic is undecidable, but we can increment it until we manage to reduce M^k to **0**: this yields a semi-decision procedure.

This leaves the problem of the search for refuted instances. The nice thing about BDDs is that they readily contain all the structure we need to build in a wealth of various proof-search methods, i.e. techniques for finding instances of a BDD that reduce it to **0**.

5.1 Making all 1-paths unsatisfiable

Take a BDD Φ (not necessarily ordered or reduced), typically representing M^k. The only way we can choose σ such that $\Phi\sigma$ reduces to **0** (we say then that σ is a *refuting substitution*) is by having σ unify enough atomic subformulæ of Φ. This can be seen on the **1**-paths: if $\Phi\sigma$ reduces to **0**, then all of its **1**-paths are propositionally unsatisfiable. These **1**-paths can be regarded as conjunctions of literals, and are therefore unsatisfiable iff σ unifies two complementary (i.e, with opposite signs) literals.

This approach is reminiscent of tableau methods (13), the connection method (2) or the method of matings (1). Indeed, if we do not consider γ and δ-rules, which govern the treatment of quantifiers (extension/amplification and Skolemization in connections/matings; or restricted extension in free-variable tableaux), α and β-rules essentially build disjunctive normal forms, trying to close branches — the analogue of **1**-paths — on the fly. Therefore, BDDs are actually complete expansions of tableaux with lemmata (20). The difference is that tableaux expand on demand, whereas BDDs are built entirely in memory. Because of this tight correspondence between tableaux and BDDs, BDDs provide a more compact representation of tableaux (22; 21).

Conceptually, and quite roughly, the previous method looks like this, when trying to find a propositionally unsatisfiable instance of an ROBDD Φ:

(1) initialize σ to the empty substitution []. For each **1**-path in Φ:
(2) choose two complementary literals A and $-A'$ on the path.
(3) unify $A\sigma$ and $A'\sigma$, yielding σ'. If it fails, the procedure fails.
(4) otherwise, set σ to $\sigma\sigma'$, and go back to step 2.

This is a non-deterministic procedure, where failure is normally implemented as backtracking (depth-first search). It would be silly to expand paths fully and then traverse the space of all paths. But traversing the space of *partial* paths, that is, paths coming down from the root but not necessarily reaching a leaf, and extending paths when necessary, yields much less paths to close, just like in tableaux or in the connection method. A pseudo-Prolog code to achieve this is:

```
close((A → B;C),Path) :- (memberunify(¬A,Path);close(B,[A|Path])),
                         (memberunify(A,Path);close(C,[¬A|Path])).
close(LEAF,_) :- LEAF=0 ; extend(Path)
```

where the second parameter of `close/2` is the current branch in the tableau (initially `[]`) and `memberunify` denotes membership with sound unification. `extend/1` is used to build extension steps in the procedure itself. In quantifier-free BDDs, this means that we continue descending from the root of a variant of the BDD we are currently considering. This means expanding the conjunction of Proposition 10 by one step (incrementing k).

To narrow the search space, we can represent quantifiers in BDDs (20; 21). The idea is borrowed from γ-formulæ in semantic tableaux; we extend Definition 2 by:

(3) *if* $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathcal{BDD}$, *then* $((\forall x \cdot \mathcal{A}) \to \mathcal{B};\mathcal{C}) \in \mathcal{BDD}$.

This yields nested BDDs, as shown on the right-hand side of Fig. 2. When we descend such nested BDDs, we do not enter the subBDDs at first, but use them for extensions in paths that contain non-negated occurrences of them. This reflects the γ-expansion rule of tableaux. In Prolog, this is best implemented by iterative-deepening depth-first search, which can in turn be compiled to give quite efficient search methods (20).

5.2 Eliminating 1-paths

A similar approach consists in not closing, but *eliminating* all 1-paths in succession. For this, we don't keep Φ static during the whole search, but reduce it w.r.t. the equivalences between atoms induced by the unifiers found, thus reducing the number of 1-paths. This already induces a change of representation, as we now need to work with ROBDDs.

We first need a few new concepts. If $\mathcal{A}$ is a set of atoms, any substitution σ induces an equivalence relation $\cong_\sigma$ on $\mathcal{A}$ by $A \cong_\sigma A'$ iff $A\sigma = A'\sigma$ (i.e, A unifies with A' through σ). Then, each equivalence relation $\cong$ on $\mathcal{A}$ is either induced by no substitution, or there is a most general substitution $\sigma(\cong)$, such that all substitutions inducing $\cong$ are instances of it. We call $\sigma(\cong)$ the *most general unifier* of $\cong$. Usually, $\mathcal{A}$ will be the set of atoms in a BDD Φ. The existence and uniqueness of $\sigma(\cong)$ is a trivial generalization of Robinson's theorem on the existence and uniqueness of most general unifiers (23).

To reduce $\Phi\sigma$ given a substitution σ, we can dispense with the rewriting of atoms A of Φ into $A\sigma$ followed by propositional reduction: choose a canonical atom in each equivalence class modulo $\cong_\sigma$, rewrite each atom A in Φ by the canonical representative of its class, and reduce the new BDD. This way, we keep applications of σ implicit. It is more interesting in practice to always rewrite every atom A in the smallest A' (the highest in the BDD) such that $A \cong_\sigma A'$. We write this A' $A \uparrow$.

The scheme of Section 5.1 is then changed to the following, where the ROBDD Φ evolves during the course of computation:

(1) initialize σ to the empty substitution []. While there exists a 1-path in Φ:
(2) choose two complementary literals A and $-A'$ on the path.
(3) unify $A\sigma$ and $A'\sigma$, yielding σ'. If it fails, the procedure fails.
(4) otherwise, set σ to $\sigma\sigma'$, replace Φ by the reduced version of Φ where each A has been replaced by $A\uparrow$, and go back to step 2.

Compare this with the previous scheme. Previously, we basically found a clever way of traversing the space of all 1-paths (by factoring them through their common prefixes as partial paths) in a fixed order ("for each" in item (1)). We now have a procedure that eliminates 1-paths in an arbitrary order: it is simply asked to choose a 1-path and to eliminate it, until none remains.

We consider full 1-paths, instead of partial ones as previously, which seems impractical at first. But in this scheme, the 1-path chosen at step 1 is effectively eliminated from the new ROBDD built at step 4. As the number of paths is exponential in the number of atoms, and each rewriting in step 4 rewrites at least one atom, then on average each pass through step 4 eliminates an exponential number of 1-paths. Moreover, the reduction in step 4 takes time polynomial in the number of nodes of Φ, which — this is the whole point in BDDs — is usually much less than the number of **1**-paths. However, we pay for doing more in one inference than in Section 5.1 with a lower inference rate.

This approach (15) differs from tableaux, connections, and matings in that the underlying formula Φ evolves during the proof search: the test for propositional unsatisfiability works incrementally, as new parts of the final substitution are found.

To reach completeness, as before, we have to incorporate an extension (or amplification) rule, that modifies the current BDD Φ by taking it in conjunction with a new copy $M\rho_{k+1}$ of the initial body M (incrementing k). Again, we can achieve this by iterative deepening depth-first search. In general, we just need a search method that interleaves extensions and searches for instantiation fairly.

5.3 Eliminating atomic subformulæ

The method in Section 5.2 can be seen as identifying a particular obstacle to unsatisfiability in Φ (the existence of **1**-paths) that we strive to eliminate. But a propositionally unsatisfiable ROBDD not only has no **1**-paths, but sports no atoms either. So, we can try to eliminate *atoms*, viewed as obstacles to propositional unsatisfiability of ROBDDs.

How do we eliminate an atom A from an ROBDD Φ? There are two cases: either there is a refuting substitution σ (one such that $\Phi\sigma$ reduces to $\mathbf{0}$) that unifies A with no other atom of Φ (we say that A is *unnecessary*); or all refuting substitutions are instances of the most general unifier of A with some other atom A' in Φ. Only atoms A' complementary to A in Φ are useful, i.e. atoms complementary on some **1**-path of Φ.

In the first case, assuming A is unnecessary, and $\Phi\sigma$ reduces to $\mathbf{0}$, then both $\Phi[\mathbf{1}/A]\sigma$ and $\Phi[\mathbf{0}/A]\sigma$ reduce to $\mathbf{0}$ (since the truth value of $A\sigma$ is independent of the truth values of all other substituted atoms), so, writing $\exists A \cdot \Phi$ for the reduced OBDD for $\Phi[\mathbf{1}/A] \vee \Phi[\mathbf{0}/A]$, $(\exists A \cdot \Phi)\sigma$ must reduce to $\mathbf{0}$. Conversely, if $(\exists A \cdot \Phi)\sigma$ reduces to $\mathbf{0}$, as Φ propositionally entails $\exists A \cdot \Phi$, then $\Phi\sigma$ reduces to $\mathbf{0}$. To eliminate atoms instead of **1**-paths, we do:

(1) initialize σ to $[]$. While there is an atom A in the ROBDD Φ:
(2) either replace Φ by $\exists A \cdot \Phi$, and go back to step 2; or choose a most general unifier σ' between A and some complementary atom A' in Φ.
(3) set σ to $\sigma\sigma'$, replace Φ by the reduced version of Φ where each A has been replaced by $A\uparrow$, and go back to step 2.

At step 2, if p is the number of distinct most general unifiers σ' of the atom A has with other atoms in Φ, we have to choose between $p + 1$ different actions: p actions where we instantiate σ by the p different σ' and reduce Φ accordingly, plus replacing Φ by $\exists A \cdot \Phi$. The procedure must choose non-deterministically, typically by backtracking.

Some points in this procedure are akin to resolution, or more precisely, to V-resolution (8), where variables can only be instantiated once (to get more, increase k). The resolution rule is used both as a propositional simplification tool, which we do not need here since ROBDDs are enough, and as a way of eliminating unnecessary atoms, by building resolvents from which these atoms have been eliminated. The **0**-paths (analogous to clauses) in $\exists A \cdot \Phi$ are just the concatenations of the **0**-paths of Φ going through $+A$ with those going through $-A$, with A excluded, as Section 4.1 has shown. Computing $\exists A \cdot \Phi$ means computing the ROBDD of all resolvents of clauses in Φ by resolving on A. All other deducible clauses are produced by the instantiation of Φ by substitutions, found by looking at unifiers of A with other atoms.

The similarity with resolution also appears in special cases of the procedure. For instance, if $p = 0$, i.e, the atom A unifies with no complementary atom in Φ, then we only have the choice of eliminating A, i.e, transforming Φ into $\exists A \cdot \Phi$. If A only occurred positively (resp. negatively) on **1**-paths in Φ, $+A$ (resp. $-A$) would be a *pure literal.* In the general case, replacing Φ by $\exists A \cdot \Phi$ is a generalization of pure literal elimination: replacing Φ by $\exists A \cdot \Phi$ means eliminating pure (non-unifiable) atoms.

In previous methods, the choice of **1**-paths was arbitrary, here the choice of atoms A is arbitrary, too. Pure atoms should be chosen first, since they reduce the size of Φ

without introducing any non-determinism. In general, A is chosen through the use of a *selection function*, just as in the linear strategy for resolution (17). The latter, as well as set-of-support strategies, can also be adapted to the BDD case, and new techniques like instance subtraction or information control are easily implemented on BDDs (16).

6 Conclusion

We showed how BDDs can be used in automated deduction. BDDs are a very rich structure, on which several viewpoints are possible, yielding as many different ways of looking for refuting substitutions; BDDs are related to well-known proof methods, like tableaux (Section 4) or resolution (Section 5.3). We firmly believe that BDDs are not only a useful propositional tool, but a promising structuring device and implementation technique in first-order logic. We hope that this paper will contribute to bringing research in BDDs and automated deduction closer together. Both areas would undoubtedly benefit.

References

1. P.B. Andrews. Theorem proving via general matings. *J. ACM*, 28(2), 1981.
2. W. Bibel. *Automated Theorem Proving*. Vieweg, 1987.
3. G. Boole. *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Dover, NY, 1958. (1st Edition 1854).
4. K.S. Brace, et. al. Efficient implementation of a BDD package. In *27^{th} DAC*, 1990.
5. R.Y. Bryant. Graph–based algorithms for Boolean function manipulation. *IEEE Trans. Computers*, C–35, 1986.
6. R.Y. Bryant. Symbolic boolean manipulation with ordered BDDs. Tech. rep., CMU, 1992.
7. J.R. Burch, et. al. Symbolic model checking: 10^{20} states and beyond. In *5th LICS*, 1990.
8. C.-L. Chang and R.C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science Classics. Academic Press, 1973.
9. A. Church. *Introduction to Mathematical Logic*, vol. 1. Princeton Univ. Press, 1956.
10. O. Coudert. *SIAM : Une Boîte à Outils Pour la Preuve Formelle de Systèmes Séquentiels*. PhD thesis, ENST, Paris, Oct. 1991.
11. M. d'Agostino. Are tableaux an improvement on truth–tables? *J. Logic, Language and Information*, 1(3), 1992.
12. E.A. Emerson. *Temporal and Modal Logic*, chapter 16. Elsevier, 1990.
13. M.C. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.
14. M.R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, 1979.
15. J. Goubault. Syntax independent connections. In D. Basin *et al.*, editora, *2nd Tableaux Workshop*, number MPI-I-93-213. Max Planck Institut für Informatik, 1993.
16. J. Goubault. Proving with BDDs and control of information. In *CADE-12*, 1994.
17. R. Kowalski and D. Kuehner. Linear resolution with selection function. *AI*, 2, 1971.
18. J.C. Madre and O. Coudert. A logically complete reasoning maintenance system based on a logical constraint solver. In *12th IJCAI*, 1991.
19. K.L. McMillan. *Symbolic Model Checking: an Approach to the State Explosion Problem*. PhD thesis, CMU, 1992.
20. J. Posegga. Deduktion mit Shannongraphen für Prädikatenlogik erster Stufe. Infix-Verlag, FRG, 1993.
21. J. Posegga and P. H. Schmitt. Automated deduction with Shannon graphs. submitted, 1994.
22. Joachim Posegga. Compiling proof search in semantic tableaux. In *7^{th} ISMIS*, LNAI, Trondheim, Norway, 1993. Springer.
23. J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1), 1965.
24. C.E. Shannon. A symbolic analysis of relay and switching circuits. *AIEE Trans.*, 67, 1938.

A Possibility-Based Propositional logic of Conditionals

Yen-Teh Hsia
Department of Information and Computer Engineering
Chung Yuan Christian University
Chung-Li, Taiwan 32023, R.O.C.

Abstract. We describe a system for propositional reasoning that is based on the idea of Gärdenfors, Makinson and Rott. Our logic turns out to be equivalent to the logic of counterfactuals of Lewis. But the resemblance stops here, i.e., at the level of the underlying formal system. Starting with the comparative possibility of Lewis, we define a notion of conditional. Formally, a conditional $\alpha \Rightarrow \beta$ (read as "given α, believe β") is just an abbreviation of a formula. But what we do here is to use the system of Lewis (or rather, our equivalent formulation of it) for reasoning with conditionals rather than comparative possibilities. Since possibility measures are the only functions from the set of all propositions to the unit interval [0, 1] that are compatible with the axioms of our logic, reasoning with conditionals in our system amounts to qualitative possibilistic reasoning. This formal correspondence allows us to use a natural method for devising proofs.

1 Introduction

How can we make rational inferences when we are given some information (e.g., Penguin $\wedge$ Bird) about a particular situation? In answering this question, Gärdenfors and Makinson [94] proposed a reasoning approach. This approach starts with the definition of *expectation ordering*. We will not use this definition here. Instead, we will use a dual notion called *possibility ordering* [Fariñas del Cerro et al. 94] in order to remain uniform in our notation.[1] Gärdenfors and Makinson's reasoning approach may be described as follows. First, we set up two sets Γ and Δ of formulas; Γ is a set of hard, non-defeasible constraints (e.g., Γ = {Penguin $\supset$ Bird}); Δ includes Γ and all of our soft, defeasible expectations (e.g., $\Delta = \Gamma \cup$ {Bird $\supset$ Fly, Penguin $\supset \neg$Fly, ...}),[2] and is closed under logical consequence. Next, we assume that there exists an ordering $\geq_P$ of *all* sentences. $\geq_P$ is called a possibility ordering, and satisfies the following axioms [Fariñas del Cerro et al. 94]. (Below, α, β, and γ are propositional formulas, while $\vdash$ is the usual propositional provability symbol.)[3]

(P1) If $\alpha \geq_P \beta$ and $\beta \geq_P \gamma$, then $\alpha \geq_P \gamma$ (Transitivity).

(P2) If $\Gamma \vdash \alpha \supset \beta$, then $\beta \geq_P \alpha$ (Dominance).

(P3) ($\alpha \geq_P \alpha \vee \beta$) or ($\beta \geq_P \alpha \vee \beta$) (Disjunctiveness).

[1] An ordering $\alpha \geq_E \beta$ of Gärdenfors and Makinson is translated into $\neg\beta \geq_P \neg\alpha$ here.

[2] We are assuming that some extraordinary penguins do fly.

[3] Gärdenfors and Makinson [94] use $\vdash$ in a different way. While we write $\Gamma \cup \{\alpha\} \vdash \beta$ here, they would write it as $\alpha \vdash \beta$, making Γ implicit.

$\alpha \geq_P \beta$ means α is *at least as possible as* β (or β is *at least as surprising as* α), and (P1) - (P3) imply that $\geq_P$ is a complete ordering (i.e., $\alpha \geq_P \beta$ or $\beta \geq_P \alpha$). We write $\alpha >_P \beta$ to mean that $\alpha \geq_P \beta$ and not $\beta \geq_P \alpha$ (i.e., α is strictly more possible than β), and we write $\alpha =_P \beta$ to mean that $\alpha \geq_P \beta$ and $\beta \geq_P \alpha$ (i.e., α is as possible as β). One way to understand (P3) is this: if α and β have numerical possibility values a and b, respectively, then by (P2) and (P3), the value of $\alpha \vee \beta$ is max(a, b). (P1) and (P2) imply that for any α in Γ and any β, $\top \geq_P \beta \geq_P \bot =_P \neg\alpha$, where $\top$ and $\bot$ are the two propositional constants 'truth' and 'falsity', respectively. (P1) - (P3) also imply that $\alpha =_P \top$ or $\neg\alpha =_P \top$, meaning that one of α and $\neg\alpha$ (maybe both) must be "entirely possible". Interpreting $\top >_P \neg\alpha$ as "α is believed" (since $\neg\alpha$ is considered less possible while α is totally possible), we see that at most one of α and $\neg\alpha$ can be believed at any one time. This conforms to our everyday use of the word 'believe'.

Once we have specified Γ and Δ, and also postulated the existence of $\leq_P$, we can then make inferences from our given information α according to a principle of reasoning that is due to Rott [91] (see also [Dubois and Prade 91]).

α nonmonotonically entails γ, denoted as $\alpha \mid\!\sim_P \gamma$,
if and only if
$\Gamma \cup \{\alpha\} \cup \{\beta : \beta \in \Delta \text{ and } \alpha >_P \neg\beta\} \vdash \gamma$.

This makes $\mid\!\sim_P$ a binary relation between formulas (given Γ and $\geq_P$). Gärdenfors and Makinson [94] call it a *comparative expectation inference relation*. See [Fariñas del Cerro et al. 94] for an interesting historical perspective of this relation.

It can be shown that any nonmonotonic inference relation $\mid\!\sim$ satisfying the postulates below is a comparative expectation inference relation [Gärdenfors and Makinson 94]. With the exception of Supraclassicality and Consistency Preservation, these postulates or "patterns of nonmonotonic inferences" as it is sometimes called, are often discussed in the literature. See, for example, the works of Delgrande [87, 88], Kraus, Lehmann and Magidor [90], Goldszmidt and Pearl [92] and Boutilier [90].

If $\Gamma \vdash \alpha \supset \beta$, then $\alpha \mid\!\sim \beta$ (Supraclassicality)
If $\Gamma \vdash \alpha \equiv \beta$ and $\alpha \mid\!\sim \gamma$, then $\beta \mid\!\sim \gamma$ (Left Logical Equivalence)
If $\alpha \mid\!\sim \beta$ and $\Gamma \vdash \beta \supset \gamma$, then $\alpha \mid\!\sim \gamma$ (Right Weakening)
If $\alpha \mid\!\sim \beta$ and $\alpha \mid\!\sim \gamma$, then $\alpha \mid\!\sim \beta \wedge \gamma$ (And)
If $\beta \mid\!\sim \gamma$, then $\top \mid\!\sim \beta \supset \gamma$ (Weak Conditionalization)
If $\top \mid\!\sim \beta \supset \gamma$ and NOT($\top \mid\!\sim \neg\beta$), then $\beta \mid\!\sim \gamma$ (Weak Rational Monotonicity)
If $\alpha \mid\!\sim \bot$, then $\Gamma \vdash \alpha \supset \bot$ (Consistency Preservation)
If $\alpha \mid\!\sim \beta$, then $\alpha \wedge \beta \mid\!\sim \gamma$ iff $\alpha \mid\!\sim \gamma$ (Cumulativity)
If $\alpha \mid\!\sim \gamma$ and $\beta \mid\!\sim \gamma$, then $\alpha \vee \beta \mid\!\sim \gamma$ (Or)
If $\alpha \mid\!\sim \gamma$ and NOT($\alpha \mid\!\sim \neg\beta$), then $\alpha \wedge \beta \mid\!\sim \gamma$ (Rational Monotonicity)

Gärdenfors and Makinson [94] also suggested that in formalizing the notion of (propositional) nonmonotonic inferences, there is no need to go beyond classical propositional logic. Everything can be done *within* the framework of propositional logic. However, they did not make their logic explicit, even though they did give the basic idea, with a couple of examples such as inferring $\alpha \mid\!\sim \gamma$ and NOT $\alpha \mid\!\sim \neg\gamma$ from $\{\alpha \wedge \gamma >_P \alpha \wedge \neg\gamma\}$.

In this paper, we formalize the idea of Gärdenfors and Makinson as a propositional logic called U, and we reference the work of Dubois [86] in our formalization. As it turns out, our logic is equivalent to the logic of counterfactuals of Lewis [73]. But the resemblance stops here, i.e., at the level of the underlying formal system. Following the suggestion of Gärdenfors, Makinson and Rott, we define conditionals in terms of the *comparative possibility* (i.e., $\alpha \geq \beta$; read as "α is at least as possible as β" in this paper) of Lewis [73]. Formally, a conditional $\alpha \Rightarrow \beta$ (read as "given α, believe β") is just an abbreviation of the formula $(\alpha = \bot) \vee_P (\alpha \wedge \beta > \alpha \wedge \neg \beta)$. But what happens here is that we use the system of Lewis (or rather, our formulation of it) for reasoning with conditionals rather than comparative possibilities.

Interpreting $\alpha \geq \beta$ as a constraint $f([\alpha]) \geq f([\beta])$, it can be shown that possibility measures [Zadeh 78] are the only functions from the set of all propositions to the unit interval [0, 1] that satisfy the axioms of our logic [Dubois 86]. Therefore reasoning with conditionals in our system is in fact qualitative possibilistic reasoning. This correspondence allows us to use a simple and effective method for devising proofs.

The main contribution of our work is as follows. We give a coherent integration of the works of Dubois [86], Lewis [73], and Gärdenfors and Makinson [94]. In doing so, we also suggest a method for qualitative possibilistic reasoning.

There are other works that are related to the use of possibility ordering for making nonmonotonic inferences. Goldszmidt and Pearl [90], for example, proposed a system (called Z) where default rules and formulas are ranked by a function that is closely related to a possibility ordering [Benferhat et al. 92]. Boutilier [92] also established a connection between system Z and epistemic entrenchment [Gärdenfors 88] - a notion that is closely related to possibility ordering [Dubois and Prade 90]. Dubois and Prade [91] use one particular version of conditional possibility in defining nonmonotonic inferences, and their approach is closely related to that of Gärdenfors and Makinson [94], since $\Pi(\alpha \wedge \gamma) > \Pi(\alpha \wedge \neg\gamma)$ if and only if $1 > \Pi(\neg\gamma \mid \alpha)$. But since the reasoning approach of Dubois and Prade is based on the notion of conditioning, it takes a quite different direction in its development. Spohn [88] also uses a numerical approach for reasoning. The quantification scheme that he uses may be viewed as a tranformation of a possibility distribution π, and his conditioning scheme is (also) such that $\Pi(\alpha \wedge \gamma) > \Pi(\alpha \wedge \neg\gamma)$ if and only if $1 > \Pi(\neg\gamma \mid \alpha)$. Spohn's approach is purely quantitative and not qualitative.

The rest of this paper is organized as follows. In Section 2, we compare the axiomatization of Gärdenfors and Makinson [94] to that of Dubois [86], and argue for the acceptance of a fourth axiom $\top >_P \bot$. In Section 3, we give the propositional logic **U** which is a qualitative counterpart of the possibility theory, and relate it to Lewis' logic of counterfactuals VN. In Section 4, we define conditionals and show how we can reason with conditionals in **U**. Section 5 concludes.

2 To require $\top > \bot$ or not - an analysis of rationality

Dubois [86] also studied an ordering $\geq$ of all sentences which may be called a possibility ordering as well. This ordering satisfies the following axioms.

(B1) $\alpha \geq \beta$ or $\beta \geq \alpha$ (Comparability).

(B2) If $\alpha \geq \beta$ and $\beta \geq \gamma$, then $\alpha \geq \gamma$ (Transitivity).

(B3) $\alpha \geq \bot$.

(B4) If $\alpha \geq \beta$, then $\alpha \vee \gamma \geq \beta \vee \gamma$.

(B5) $\top > \bot$ (Nontriviality).

As it turns out, (B1) - (B4) are equivalent to (P1) - (P3) [Dubois and Prade 90]. Therefore the only difference between $\geq_P$ and $\geq$ is in the requirement that $\top > \bot$ (i.e., $\top \geq \bot$ and not $\bot \geq \top$). Now the question is: should we or should we not have this axiom in formalizing our intuitive notion of belief (or surprise)? This is what we discuss in this section.

To reason with the use of a logic that is based on classical propositional logic, we must first assume the existence of a non-empty set of propositional primitives. Let us call this set $\mathcal{P}$, and say that $\mathcal{P}$ is finite (we restrict our discussion to the finite case here). Intuitively, each element of $\mathcal{P}$ is intended for modelling some real world aspect. And so by saying that a primitive is either true or false, we mean to accept a very important assumption.

> **Assumption 2.1 (Basic assumption of reasoning with classical logics)**
> No matter what real-world aspect we are modelling (using a primitive), we can always tell whether that aspect of the world holds or not.

For example, if we use a primitive such as RAIN to mean, intuitively, whether it is raining outside. We are making the tacit assumption that we can *always* tell whether it is raining outside. This leads to the following.

> **Proposition 2.2 (Consequence of Assumption 2.1)**
> The part of the world we are interested in can always be represented by exactly one element of Θ, where Θ is the set of all mappings (called *interpretations* or *worlds* or *valuations*) from $\mathcal{P}$ to the set $\{T, F\}$.

Now let $\mathcal{L}_P$ be the least set of formulas containing $\mathcal{P}$, closed under $\neg$ and $\wedge$ (with the usual abbreviations $\vee$, $\supset$, and $\equiv$, where $\equiv$ is "if and only if"). We use $[\alpha]$ to denote the set of all models of α ($[\alpha] \subseteq \Theta$). Assumption 2.3 is often discussed.

> **Assumption 2.3 (The propositional view of formulas)**
> By saying that a formula α is *true*, we mean that the element of Θ that represents the world is in $[\alpha]$.

Not all people accept Assumption 2.3. But those who are interested in ideal or rational agents do accept it. It is in fact the main assumption underlying what Hintikka [75] refers to as *logical omniscience*.

> **Proposition 2.4 (Consequence of Assumption 2.3)**
> By saying that a formula α is *strictly more possible than* a formula β, we mean that our potential surprise in "the representation of the world is in $[\beta]$" is higher than our potential surprise in "the representation of the world is in $[\alpha]$".

Suppose we accept Assumptions 2.1 and 2.3. Then we are obliged to accept Propositions 2.2 and 2.4 as well. Thus, in theory, we could just work with subsets of Θ. Because $(2^{\Theta}, \subseteq, \cap, \cup)$, with top element Θ and bottom element $\emptyset$, and $(\mathcal{L}_P, \supset, \wedge, \vee)$, with top $\top$ and bottom $\perp$, become the same Boolean algebra once we identify every formula α in $\mathcal{L}_P$ with $[\alpha]$ in 2^{Θ}, $\neg$ with the set complement operator, $\supset$ with $\subseteq$, $\wedge$ with $\cap$, $\vee$ with $\cup$, $\top$ with Θ, and $\perp$ with $\emptyset$.

Going back to the question that we posed at the beginning of this section: should we consider $\top$ to be strictly more possible than $\perp$? Since the representation of the world *must be* in Θ (by Proposition 2.2), rationality requires that we are not surprised that it is in Θ, and that we *are* surprised that it is in $\emptyset$. $[\top] = \Theta$ and $[\perp] = \emptyset$. Therefore our potential surprise in $\perp$ should be higher than our potential surprise in $\top$ (by Proposition 2.4). This makes $\top$ strictly more possible than $\perp$ a logical necessity. That is, we should rationally consider $\top > \perp$ to be intuitively valid.

3 The propositional logic U

Suppose we now add a fourth axiom "$\top >_P \perp$" to the axioms of Gärdenfors and

Makinson [94]. The resulting axiomatization then is as follows.

(P1) If $\alpha \geq_P \beta$ and $\beta \geq_P \gamma$, then $\alpha \geq_P \gamma$; (Transitivity).

(P2) If $\Gamma \mid\!- \alpha \supset \beta$, then $\beta \geq_P \alpha$; (Dominance).

(P3) $(\alpha \geq_P \alpha \vee \beta)$ or $(\beta \geq_P \alpha \vee \beta)$; (Disjunctiveness).

(P4) $T >_P \bot$ (Nontriviality).

Clearly, we can use (P1) - (P4) to define a logic. We define such a logic (called **U**; for "unexpected") as follows.

Primitives of the language $\mathcal{L}_U$ are *comparisons* of the form $\alpha \geq_P \beta$ (read as "α is at least as possible as β"), where α and β are object-level formulas ($\alpha, \beta \in \mathcal{L}_P$). $\mathcal{L}_U$ is the least set of formulas containing all comparisons, closed under $\neg_P$, $\wedge_P$ (with the usual abbreviations $\vee_P$, $\supset_P$, $\equiv_P$). Below, we write $\geq$ instead of $\geq_P$. We also write $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$ instead of $\neg_P$, $\wedge_P$, $\vee_P$, $\supset_P$, $\equiv_P$, respectively, whenever there are no object-level connectives (i.e., $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$) appearing in a formula. The following abbreviations are then used.

$\alpha = \beta$ ("α is as possible as β") is a shorthand for $(\alpha \geq \beta) \wedge (\beta \geq \alpha)$.

$\alpha > \beta$ ("α is more possible than β") is a shorthand for $(\alpha \geq \beta) \wedge \neg(\beta \geq \alpha)$.

The axiom schemata of **U** are as follows.

1. (Tautology) Classical axioms.
2. (Transitivity) $(\alpha \geq \beta \wedge_P \beta \geq \gamma) \supset_P \alpha \geq \gamma$.
3. (Dominance) $\beta \geq \alpha$, where $\alpha \supset \beta$ is an object-level tautology (i.e., $\mid\!- \alpha \supset \beta$).
4. (Disjunctiveness) $(\alpha \geq \alpha \vee \beta) \vee_P (\beta \geq \alpha \vee \beta)$
5. (Nontriviality) $T > \bot$

The inference rule of **U** is modus ponens (from p and p $\supset_P$ q, infer q) as usual. When a formula q can be inferred from a given set Σ of premises, we denote it as $\Sigma \mid\!-_P q$. The semantics of **U** is just the usual semantics of propositional logic with the restriction that (2) - (5) be valid, as defined below.

An *interpretation* I is an assignment of a truth value (an element of {T, F}) to each of the (meta-level) propositional primitives. The way a (meta-) formula is evaluated is by using the usual truth table technique. A formula p is a *tautology* if and only if p is evaluated to T under all interpretations. A formula p is *unsatisfiable* if and only if p is evaluated to F under all interpretations. A *U-interpretation* is any interpretation that satisfies axiom schemata (2) through (5).[4] Formulas $p_1, p_2, \ldots, p_n$ *U-logically entails* a formula q, denoted as $p_1, p_2, \ldots, p_n \models_P q$, if and only if for all U-interpretations I, if $p_1, p_2, \ldots, p_n$ are true under I, then q is also true under I. A set Σ of formulas is *U-consistent* if and only if there exists an U-interpretation I under which all formulas in Σ are true (I is called a *U-model* of Σ). Soundness and completeness of **U** follow that of propositional logic.

Theorem 3.1. **U** is sound and complete (i.e., $\Sigma \mid\!-_P q$ if and only if $\Sigma \models_P q$)

It turns out that **U** is equivalent to the conditional logic VN of Lewis [73] ([Fariñas del Cerro and Herzig 91]). The axioms and inference rules of VN is this:

[4] It may be of interest to note that T (a shorthand for T = T) is evaluated to T under all interpretations, because $\alpha = \alpha$ is a tautology. In contrast, $\bot$ (a shorthand for $\bot = T$) is evaluated to F under all *E-interpretations*, because $\bot \geq T$ must be assigned F, so that $T > \bot$ is satisfied.

Axioms of VN:
(VN0) Classical axioms
(VN1) $\alpha \geq \beta \wedge \beta \geq \gamma \supset \alpha \geq \beta$
(VN2) $\alpha \geq \beta \vee \beta \geq \alpha$
(VN3) $\neg(\bot \geq \top)$
Inference rules of VN:
(VN4) modus ponens
(VN5) from $\alpha \supset (\beta_1 \vee \ldots \vee \beta_n)$, infer $(\beta_1 \geq \alpha) \vee \ldots \vee (\beta_n \geq \alpha)$

Therefore **U** is nothing but an alternative axiomatization of VN. But note that in **U**, we only need one inference rule which is good old modus ponens. We also think that it is intuitively more coherent to just work on a meta-propositional-level by requiring that all primitives be comparisons. In other words, we feel that our intuition for a usual object level formula α should actually be specified as the comparison $\neg\alpha = \bot$.

4 Reasoning about conditionals

Recall that a comparative expectation inference relation $\alpha \mid\!\sim_P \gamma$ is defined as $\Gamma \cup \{\alpha\} \cup \{\beta : \beta \in \Delta \text{ and } \alpha >_P \neg\beta\} \vdash \gamma$. But Δ, the set of all expectations or beliefs, should actually be the set $\{\beta : \top >_P \neg\beta\}$. This makes the definition of $\alpha \mid\!\sim_P \gamma$ become $\Gamma \cup \{\alpha\} \cup \{\beta : \alpha >_P \neg\beta\} \vdash \gamma$, which is equivalent to the condition $\Gamma \cup \{\alpha\} \vdash \gamma$ OR $\alpha \wedge \gamma >_P \alpha \wedge \neg\gamma$ [Gärdenfors and Makinson 94; Theorem 3.5]. The condition $\Gamma \cup \{\alpha\} \vdash \gamma$ OR $\alpha \wedge \gamma >_P \alpha \wedge \neg\gamma$ itself is equivalent to the condition $(\alpha \wedge \neg\gamma) =_P \bot$ OR $\alpha \wedge \gamma >_P \alpha \wedge \neg\gamma$, which in turn is equivalent to the condition that $\alpha =_P \bot$ OR $\alpha \wedge \gamma >_P \alpha \wedge \neg\gamma$. And so to formulate the notion of comparative expectation inference relations in **U**, we need only define a *conditional* $\alpha \Rightarrow \gamma$ as an abbreviation of the formula $(\alpha =_P \bot) \vee_P (\alpha \wedge \gamma >_P \alpha \wedge \neg\gamma)$. Once we have made $\alpha \Rightarrow \gamma$ the U-counterpart of $\alpha \mid\!\sim_P \gamma$, all postulates that Gärdenfors and Makinson [94] proposed, to be satisfied by all nonmonotonic inference relations $\mid\!\sim$, simply become theorems of **U**. That is, we can take any postulate that Gärdenfors and Makinson [94] proposed (e.g., the postulate that if $\Gamma \vdash \alpha \equiv \beta$ and $\alpha \mid\!\sim \gamma$, then $\beta \mid\!\sim \gamma$), and translate it into **U** by replacing every occurence of the symbol $\mid\!\sim$ by the symbol $\Rightarrow$, and also replacing every formula α that is deducible at the object level by the formula $\neg\alpha = \bot$. What we get then is a theorem of **U** (e.g., the formula $(\neg(\alpha \equiv \beta) = \bot) \wedge_P (\alpha \Rightarrow \gamma) \supset_P (\beta \Rightarrow \gamma)$ is a theorem). Note that Consistency Preservation (formulated as $(\alpha \Rightarrow \bot) \supset \alpha = \bot$ in **U**) does not hold for many kinds of nonmonotonic inference, including preferential reasoning that are either classical, stoppered or ranked [Makinson and Gärdenfors 90].

Below, we list some other theorems and non-theorems that may be of interest. But first, let us formally introduce our last two abbreviations.

α ("α is categorically believed") is a shorthand for $\neg\alpha = \bot$.
$\alpha \Rightarrow \beta$ ("given α, believe β") is a shorthand for $(\alpha = \bot) \vee_P (\alpha \wedge \beta > \alpha \wedge \neg\beta)$.

Theorem 4.1. (Theorems of **U**)
(1) (Embedded object-level modus ponens) $\{\alpha, \alpha \supset \beta\} \vdash_P \beta$
(2) (The penguin triangle) $\{B \Rightarrow Fly, P \Rightarrow B, P \Rightarrow \neg Fly\} \vdash_P B \wedge P \Rightarrow \neg Fly$
(3) (Conditional Chaos) $\vdash_P (\alpha = \bot) \supset \alpha \Rightarrow \gamma$
(4) (Conditional Transitivity)
$\vdash_P ((\alpha \Rightarrow \beta) \wedge \neg(\beta \Rightarrow \neg\alpha)) \wedge ((\beta \Rightarrow \gamma) \wedge \neg(\gamma \Rightarrow \neg\beta)) \supset ((\alpha \Rightarrow \gamma) \wedge \neg(\gamma \Rightarrow \neg\alpha))$

Theorem 4.2. (Non-theorems) The following are not theorems of **U**.
(1) (Difference in implications) $(\neg\alpha = \bot \supset_P \neg\beta = \bot) \supset_P (\neg(\alpha \supset \beta) = \bot)$
(2) (The Nixon diamond) $(R \Rightarrow \neg P) \wedge_P (Q \Rightarrow P) \supset_P (R \wedge Q \Rightarrow P) \vee_P (R \wedge Q \Rightarrow \neg P)$

5 A Natural Proof-Finding Method

U is just an instance of classical propositional logic. Therefore we can use any propositional theorem prover that is available to us to test the "theoremhood" of any formula of **U**. This is the main advantage of formulating our notion of nonmonotonic reasoning *within* the framework of propositional logic. There is one minor problem, however: the resulting (mechanical) proofs that we get may be quite incomprehensible. In other words, with the use of a mechanical theorem prover, we get to know whether a formula is theorem or not, but we may not quite understand why. Moreover, it can be very difficult for us humans to try to prove or disprove the theoremhood of any formula by ourselves, for we have little intuitions to work with. Thus, it is desirable to find a natural way of devising proofs. Below, we describe such a method.

Suppose we interpret $\alpha \geq_P \beta$ as a constraint $f([\alpha]) \geq f([\beta])$ that must be satisfied by a function f. Then, among all functions from 2^{Θ} to [0, 1],[5] only possibility measures satisfy the requirements (P1) - (P4);[6] moreover, a strictly agreeing possibility measure always exists [Dubois 86]. And so if we assume that there is a unique possibility measure Π representing some state of mind, then we can reason about this state of mind Π without ever mentioning any specific numbers; in particular, reasoning about $\alpha \Rightarrow \gamma$ in **U** will be exactly the same as reasoning about the property $\Pi(\alpha) = \Pi(\bot) = 0$ OR $\Pi(\alpha \wedge \gamma) > \Pi(\alpha \wedge \neg\gamma)$. What this means then is the following. To find any proof by ourselves (i.e., without using a propositional theorem prover), we can first work within the setting of the possibility theory, reasoning about properties such as $\Pi(\alpha) = 0$ OR $\Pi(\alpha \wedge \gamma) > \Pi(\alpha \wedge \neg\gamma)$; and then, we just translate our informal reasoning into **U**. This is a very natural way for finding proofs, because there is a possibility distribution π we can assume (and use).

Take the penguin triangle as an example. From $B \Rightarrow Fly$ (i.e., $B = \bot \vee_P B \wedge Fly > B \wedge \neg Fly$) and $P \Rightarrow B$ (i.e., $P = \bot \vee_P P \wedge B > P \wedge \neg B$) and $P \Rightarrow \neg Fly$ (i.e., $P = \bot \vee_P P \wedge \neg Fly > P \wedge Fly$), we need to show that $B \wedge P \Rightarrow \neg Fly$ (i.e., $B \wedge P = \bot \vee_P B \wedge P \wedge \neg Fly > B \wedge P \wedge Fly$). To find such a proof, we can partition Θ, the set of all interpretations, into the set {a, b, c, d, e, f, g, h} below. Then, we just deduce properties of Π using, for example, "abd" as a shorthand for "$\max\{\pi(a), \pi(b), \pi(d)\}$".

[5] f is assumed to be such that $f(\Theta) = 1$ and $f(\varnothing) = 0$.

[6] A *possibility measure* Π is a function from 2^{Θ} to [0, 1], satisfying the conditions that (1) $\Pi(\Theta) = 1$, (2) $\Pi(\varnothing) = 0$, and (3) $\Pi(A \cup B) = \max(\Pi(A), \Pi(B))$. From Π, we can define $\pi : \Theta \rightarrow [0, 1]$, where $\pi(\omega) = \Pi(\{\omega\})$. π is called a *possibility distribution*. It completely determines Π. And so one other way of understanding possibility theory is to postulate the existence of a possibility distribution π on Θ. π is such that there is always a world ω that is totally possible ($\pi(\omega) = 1$), while Π is induced by π in that $\Pi(A) = \max\{\pi(\omega): \omega \in A\}$. This view of possibility theory has a close analogy with probability theory (just replace maximum by addition) : a probability distribution p is such that $\sum\{p(\omega) : \omega \in \Theta\} = 1$; moreover, $P(A) = \sum\{p(\omega) : \omega \in A\}$.

B	P	Fly	abbreviation
T	T	T	a
T	T	F	b
T	F	T	c
T	F	F	d
F	T	T	e
F	T	F	f
F	F	T	g
F	F	F	h

Now, if $B = \bot$ ($abcd = 0$) or $P = \bot$ ($abef = 0$), then we are done, since $B \wedge P$ has to be $\bot$ (i.e., $ab = 0$). So we need only show that from $ac > bd$ and $ab > ef$ and $bf > ae$, we can deduce $b > a$. This is proved by contradiction. Suppose $a \geq b$. Then $a > ef$ (from $ab > ef$); and so $a \geq bef$, which contradicts $bf > ae$. This "basic idea" can always be translated into a corresponding proof in **U** at the cost of introducing some tedious detail. The following is an illustration.

	Basic proof idea	Corresponding proof statements
Case 1:	$abcd = 0$ or $abef = 0$	$B = \bot \vee_P P = \bot$
	$ab = 0$	$B \wedge P = \bot$ (since $B \geq B \wedge P \wedge_P P \geq B \wedge P$)
Case 2:	$ac > bd$	$B \wedge Fly > B \wedge \neg Fly$
	and $ab > ef$	$\wedge_P\ P \wedge B > P \wedge \neg B$
	and $bf > ae$	$\wedge_P\ P \wedge \neg Fly > P \wedge Fly$
	Suppose $a \geq b$	Suppose $B \wedge P \wedge Fly \geq B \wedge P \wedge \neg Fly$
	$a > ef$	$B \wedge P \wedge Fly > P \wedge \neg B$ (since $B \wedge P \wedge Fly \geq B \wedge P$)
	$a \geq bef$	$B \wedge P \wedge Fly \geq B \wedge P \wedge \neg Fly$
		$\wedge_P\ B \wedge P \wedge Fly \geq \neg B \wedge P \wedge Fly$
		$\wedge_P\ B \wedge P \wedge Fly \geq \neg B \wedge P \wedge \neg Fly$
	contradicts with $bf > ae$.	contradicts with (three statements) $(B \wedge P \wedge \neg Fly \geq P \wedge \neg Fly)$ $\vee_P\ (\neg B \wedge P \wedge \neg Fly \geq P \wedge \neg Fly)$, $P \wedge \neg Fly > P \wedge Fly$, and $P \wedge Fly \geq B \wedge P \wedge Fly$.

As another example, consider the rule Conditional Transitivity. It can be written as $\alpha ==> \beta \wedge_P \beta ==> \gamma \supset_P \alpha ==> \gamma$, where $\alpha ==> \beta$ is a shorthand for the formula $\alpha \Rightarrow \beta \wedge_P \neg_P(\beta \Rightarrow \neg \alpha)$, and α and β are usual propositional formulas. The intuition underlying Conditional Transitivity is that a formal notion of transitivity does hold; however, whenever we express the idea that "given α, believe β", we should also make sure that given β, we will not believe $\neg \alpha$. To prove this rule, again we partition Θ into the set $\{a, b, c, d, e, f, g, h\}$ ("a" is $\alpha \wedge \beta \wedge \gamma$, "b" is $\alpha \wedge \beta \wedge \neg\gamma$, ..., "h" is $\neg\alpha \wedge \neg\beta \wedge \neg\gamma$). The given premises are : (i) $ab > cd$ or $abcd = 0$, (ii) $ab \geq ef$ and $abef \neq 0$, (iii) $ae > bf$ or $abef = 0$, and (iv) $ae \geq cg$ and $aceg \neq 0$. We need to show that (I) $ac > bd$ or $abcd = 0$, and (II) $ac \geq eg$ and $aceg \neq 0$. Note that $abcd = 0$ contradicts with (ii), while $abef = 0$ contradicts with (iv). And so we only need to show that (I) and (II) follow from $ab > cd$, $ab \geq ef$, $abef \neq 0$, $ae > bf$, $ae \geq cg$, and $aceg \neq 0$. Suppose $ac > bd$ is false, i.e., $bd \geq ac$. $b > d$ (since $d \geq b$ contradicts with $bd \geq ac$ and $ab > cd$). And so, $b \geq ac$, making $b \geq ef$ (since $ab \geq ef$). Thus, $b \geq acef$. But this contradicts with $ae > bf$. Therefore $ac > bd$, i.e., (I) is proved. Next, suppose $ac \geq eg$ is false, i.e., $eg > ac$. $e \geq g$ (since $g > e$ contradicts with $eg > ac$ and $ae \geq cg$). And so $e > ac$. $e > bf$ (since ae > bf). Thus, e > abcf, which contradicts with ab ≥ ef. Therefore ac ≥ eg, i.e., the first half of (II) is proved. Finally, aceg ≠ 0 (the second half of (II)) is a premise.

6 Discussion and conclusion

In a word, the main contribution of this work is *integration*. We first strengthened the axiomatization of Gärdenfors and Makinson [94] in view of the work of Dubois [86]. Then, we formalized the idea of Gärdenfors and Makinson [94] as a purely propositional logic called **U**, making usual propositional reasoning a special case of it, and we also related **U** to the logic of Lewis [73]. In retrospect, we feel that Lewis [73] had the right intuition in formalizing the notion of comparative possibility, but he lacked the right reasoning mechanism which Rott [91] proposed eighteen years later. Gärdenfors and Makinson [94], on the other hand, paved the way for formally integrating the work of Lewis [73] and Rott [91], but they stopped short of formalizing the whole thing which is **U** - the system that we introduced in this paper.

One might be suspicious about **U**. After all, it is *just an instance of classical propositional logic*, while many conditional logics of its kind have gone beyond (pure) propositional logic in order to be of interest (see, for example, [Boutilier 90; Delgrande 87, 88; Goldszmidt and Pearl 92; Kraus, Lehmann and Magidor 90]). Is it perhaps the case that **U** is not so interesting? We think not. Look at what this system can do. It supports all of the rational inference patterns (as theorems) we listed in Secion 1, plus Conditional Transitivity. This is already more powerful than, say, system **P** of Kraus, Lehmann and Magidor [90]. Unlike many other systems (including Lewis' system VN), there is no additional inference rule in **U**; only modus ponens. Also, with **U**, we can perform object-level reasoning (since object-level formulas α are expressible as $\neg\alpha =_P \bot$) on the meta-level. These are all nice features in our opinion. For the moment, we do not think **U** is completely investigated. There may be other interesting inference patterns like Conditional Transitivity that we have not discovered. **U** can be useful in this respect, that is, we can experiment with it to see what other interesting inference patterns are there.

Apart from introducing **U** and formulating $\alpha \mid\!\sim_P \beta$ as $\alpha \Rightarrow \beta$ in **U**, we have also suggested a more natural way for finding proofs ourselves. To appreciate our method, the reader is encouraged to try to prove Conditional Transitivity using a related system such as the set of postulates for nonomonotonic inference relations proposed by Gärdenfors and Makinson [94] (see Section 1).

Throughout this paper, we have only referenced the work of Fariñas del Cerro, Herzig, and Lang. It is now a good time to count their contributions and clarify the relation between their work and ours. In 1991, Fariñas del Cerro and Herzig described a system called QPL, and pointed out its equivalence to VN [Lewis 73]. QPL is an equivalent formulation of **U**, and is based on (B1) - (B5) (see Section 2). In a more recent work, Fariñas del Cerro, Herzig, and Lang [94] extend the work of Gärdenfors and Makinson [94]. Specifically, they start with a set E of inequalities of the form either $\alpha \geq \beta$ or $\alpha > \beta$. Then, they define $\alpha \mid\!\sim_E \beta$ to be a relation that holds if and only if $\alpha \mid\!\sim_P \beta$ holds for *every* possibility ordering satisfying E. This makes $\mid\!\sim_E$ a preferential inference relation [Kraus et al. 90]. Unfortunately, $\mid\!\sim_E$ satisfies neither Weak Rational Monotonicity nor Rational Monotonicity. Thus, while we try to "complete" the work of Gärdenfors and Makinson [94], Fariñas del Cerro, Herzig, and Lang [94] try to extend the work of Gärdenfors and Makinson [94]. And intuitively, we feel that their proposed extension is incompatible with our proposed completion.

References

Benferhat, S., Dubois, D. and Prade, H. (1992). Representing default rules in possibilistic logic. *Proceedings of the Third International Conference on Knowledge Representation and Reasoning* (KR-92), Cambridge, Massachusetts, 637-684.

Boutilier, C. (1990). Conditional logics of normality as modal systems. In Proceedings of AAAI-90, Boston, MA, 594-599.

Boutilier, C. (1992). Conditional logics for default reasoning and belief revision, Ph.D. Thesis, University of Toronto, Toronto, Ontario.

Delgrande, J.P. (1987). A first-order logic for prototypical properties. *Artificial Intelligence* **33**, 105-130.

Delgrande, J.P. (1988). An approach to default reasoning based on a first-order conditional logic: revised report. *Artificial Intelligence* **36**, 63-90.

Dubois, D. (1986). Belief structures, possibility measures and decomposable set-functions. *Computers and Artificial Intelligence*, Bratislava **5**, 403-416.

Dubois, D. and Prade, H. (1990). Epistemic entrenchment and possibilistic logic. *Artificial Intelligence* **50**, 223-239.

Dubois, D. and Prade, H. (1991). Possibilistic logic, preference models, non-monotonicity and related issues. In *Proceedings of IJCAI-91*, 24-30.

Fariñas del Cerro, L. and Herzig, A. (1991). A modal analysis of possibility theory. In*Symbolic and Quantitative Approaches to Uncertainty* (Kruse, R. and Siegel, P. eds.), Lecture Notes in Computer Science 548, Springer-Verlag, 58-62.

Fariñas del Cerro, L., Herzig, A., and Lang, J. (1994). From ordering-based nonmonotonic reasoning to conditional logics. *Artificial Intelligence* **66**, 375-393.

Gärdenfors, P. and Makinson, D. (1994). Nonmonotonic Inference Based on Expectations. *Artificial Intelligence* **65**, 197-245.

Goldszmidt, M. and Pearl, J. (1990). On the relation between rational closure and System Z. In *Proceedings of third International Workshop on Nonmonotonic Reasoning*, South Lake Tahoe, CA, 130-140.

Hintikka, J. (1975). Impossible possible worlds vindicated. *Journal of Philosophical Logic* 4, 475-484.

Kraus, S., Lehmann, D. and Magidor M. (1990). Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* **44**, 167-207.

Lewis, D.K. (1973). *Counterfactuals*. Harvard University Press.

Makinson, D. and Gärdenfors, P. (1990). Relations between the logic of theory change and nonmonotonic logic. *Proceedings of the RP2 First Workshop DRUMS*, Albi, France, 171-190.

Rott, H. (1991). Two methods of constructing contractions and revisions of knowledge systems. *Journal of Philosophical Logic* **20**, 149-173.

Spohn, W. (1988). A general non-probabilistic theory of inductive reasoning. *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, Minnesota, 315-322.

Zadeh, L.A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **1**, 3-28.

Incremental Processing of Logic Database Relations

Yan-Nong Huang, Veronica Dahl, Jiawei Han

School of Computing Science, Simon Fraser University
Burnaby, B.C. V5A-1S6, Canada
electronic mail : {ynhuang, veronica, han}@cs.sfu.ca

Abstract. In this paper, we study a new evaluation technique for incrementally maintaining derived facts whenever updates occur on base facts. Our method is based upon the semi-naive evaluation so it is easy to implement; it is set-oriented, multiple updates may be dealt with in a single manipulation; and insertion and deletion are treated uniformly.

Keywords: Intelligent-Information-Systems, Evolutionary-Computation, Logic-Database, Fact-Update, Incremental-Evaluation

1 Introduction

Given a logic database, one may calculate derived facts in advance and store (or materialize) them. Thus, the response time can be expected to be sped-up. In this paper, we study a new evaluation technique for incrementally maintaining derived facts whenever updates occur on base facts.

Within the incremental maintenance, we consider only proofs involving at least one updated axiom; furthermore, the proof reconstruction uses as many as possible previously proven lemmas. More surprisingly, we store for each theorem only the number of its proofs rather than the proof structures; hence, a theorem is valid if and only if its proof number is greater than 0. This may considerably reduce the storage space and improve the processing efficiency.

Our method is set-oriented, multiple updates may be dealt with in a single manipulation; and insertion and deletion are treated uniformly.

We here assume no cyclic proof occurs in the derivation. As a matter of fact, other strategies are also possible to prevent from looping. For example, we may consider only theorems whose proof lengths are not greater than some positive integer. It is noted that under the proof length assumption, our method is applicable to any Horn clauses, including those involving function symbols.

The structure of the paper is as follows. After a brief revision of the semi-naive evaluation algorithm in section 2, Sections 3 and 4 study respectively theorem generation and maintenance algorithms. Finally, we give our concluding remarks. Correction proofs as well as detailed discussions are omitted from the present version of the paper, [10] provides a full description of the proposed approach.

2 Preliminary

Recall that a logic database comprises an extensional database (EDB) and an intensional database (IDB). Here we review a well-known recursive data processing method: semi-naive evaluation. For a more complete presentation on Logic Programming and Deductive Databases, one can for example refer to [6] and [11].

As its name implies, the basic idea of this algorithm is quite 'naive'. We begin with the set of axioms, by applying the derivation rules, obtain the theorems of the first 'layer'; then take these theorems as new starting point, by the application of derivation rules, to derive the theorems of the second 'layer'; and so on. Generally, to derive the theorems of next 'layer', theorems just produced must be used. This process terminates when no more new theorems can be generated.

The semi-naive evaluation follows.

Semi-Naive Algorithm
Input: $DB = EDB \cup IDB$: a logic database;
Output: Y: semantics of DB;
Method:

```
Begin
1) $\Delta Y := EDB$;
2) $Y := \emptyset$;
3) while $\Delta Y \neq \emptyset$ do
4)      $\Delta Y_{pivot} := diff\_I(Y, \Delta Y)$;
5)      $Y := Y \cup \Delta Y$;
6)      $\Delta Y_{pivot} := \Delta Y_{pivot} - Y$;
7)      $\Delta Y := \Delta Y_{pivot}$;
8) endwhile;
End;
```

Macro $diff\_I()$ used in sentence 4) is a differential version of the immediate consequence operator, which makes the essential distinction between the naive and semi-naive evaluation algorithms (To note that the above algorithm is slightly different from the versions usually used in the literature, e.g [11]).

$$diff\_I(Y, \Delta Y) = \{h\theta : h \leftarrow a_1, a_2, \ldots, a_n \in IDB, \theta \text{ is a substitution such that } \forall i \in [1, n],\ a_i\theta \in Y \cup \Delta Y \text{ and } \exists j \in [1, n] \text{ such that } a_j\theta \in \Delta Y\}$$

3 Theorem Generation

Our method is based upon the memorization of proof number for each theorem. So, we will embed the operation of counting theorem proof numbers into the process of producing theorems. Since theorem derivation procedures consider explicitly or implicitly almost all possible derivations for any theorem, it is realistic to modify these procedures in order to incorporate the theorem proofs counting operation. The theorem derivation procedure considered here is the semi-naive evaluation outlined in section 2.

Some notational conventions are first introduced. EDB is the initial extensional database. $\Delta EDB = \Delta EDB^+ \cup \Delta EDB^-$, where ΔEDB^+ and ΔEDB^- are subsets of inserted and deleted base facts. Updates under consideration are ordinary in that $EDB \cap \Delta EDB^+ = \emptyset$ and $\Delta EDB^- \subseteq EDB$.

Base facts in EDB are associated with +1; inserted and deleted facts are associated with +1, −1 respectively. For a given base fact f, c(f) denotes the counter value of f.

Now, it might be helpful to introduce a formal concept for 'proof'.

Definition 1. Let $DB = IDB \cup EDB$ be a logic database, ΔEDB be updates to EDB. A proof is a couple $< S, Q >$ where S is a tree defining the structural characteristics, Q a mapping from the set of S-leaves into {-1, 0, 1} defining the quantity property. If θ is a ground substitution; the definition of a proof structure is given inductively as follows:

1. Let p be a base predicate of arity n, $x_1, x_2, \ldots, x_n$ be n variables, then, $p(x_1, x_2, \ldots, x_n)\theta \in EDB \cup \Delta EDB$ is a proof structure.
2. If $h \leftarrow a_1, a_2, \ldots, a_n \in IDB$ and for each $i \in [1, n]$, there exists a proof tree, of which the root is $a_i\theta$. Then, the tree with $h\theta$ as root; proof trees for $a_i\theta$ ($i \in [1, n]$) as subtrees; is a proof tree.
3. There are no other proof trees. □

Let $< S, Q >$ be a proof, it is acyclic if any path on S from the root to leaf contains no two identical nodes; it is a valid proof if Q(f) = 1 for any leaf f of S. A theorem established by at least one valid proof is said to be valid as well. Recall that we here consider only acyclic proofs, this condition will not be repeated explicitly.

Notice that the above definition for proofs is slightly more general than the usual logic definition. Leaves not corresponding to EDB elements are allowed here. This treatment is useful when considering an inserted EDB fact which was not included in the initial EDB, since we might have this inserted base fact appeared in the generation algorithm's input with the counter 0. In doing so, the generation and maintenance algorithms might be expected to produce exactly the same set of proof structures, but with different combinations of leaf node counters (see Theorem 6 of Section 5). Consideration can be focused only on these counter combinations.

Let us turn back to the semi-naive evaluation presented in Section 2.

1. Due to sentence 6), if a theorem has several proofs of different lengths, then only proofs of the shortest length can be referenced in proving other theorems. Therefore, for a given theorem, not all possible proofs are likely to be traced by the algorithm.

2. When accumulating partial results, the set-theoretic operation $\cup$ is used; so, the proof numbers are not counted.

In order to be able to identify proof structures, we study here an auxiliary algorithm 'Proof Construction Algorithm' which constructs explicitly proof trees. Note that this algorithm is only used to exhibit ideas rather than for implementation. The following points are taken into consideration.

1. A supplementary field 'proof' will be added to each (base or derived) fact to store the proof structure for this fact. The attribute 'proof' is therefore to be appended to the corresponding relations.

2. Given a logic database $DB = EDB \cup IDB$, one needs to rewrite it into $DB^p = EDB^p \cup IDB^p$ to manipulate proof trees. Generally,
   - If $f = a(t_1, t_2, \ldots, t_n)$ is a base fact, then, $a(t_1, t_2, \ldots, t_n, < f, Q >) \in EDB^p$, where Q(f) = c(f).
   - If $h(Y) \leftarrow a_1(X_1), a_2(X_2), \ldots, a_n(X_n) \in IDB$ where Y, X_i are vector variables and $n > 0$, then, $h(Y, < h(Y)[p_1, p_2, \ldots, p_n], Q >) \leftarrow a_1(X_1, < p_1, Q_1 >), a_2(X_2, < p_2, Q_2 >), \ldots, a_n(X_n, < p_n, Q_n >) \in IDB^p$, where $h(Y)[p_1, p_2, \ldots, p_n]$ denotes the tree with $h(Y)$ as root, $p_1, p_2, \ldots, p_n$ as subtrees, and $Q = Q_1 \cup Q_2 \cup \ldots Q_n$.
3. Since we here consider cycle-free proofs, and intend to trace all possible proofs for a given theorem, the operation of removing previously produced theorems from ΔS (sentence 6 of semi-naive evaluation) is therefore omitted.

Similarly to EDB, ΔEDB may be augmented with proof structures to yield ΔEDB^p. That is, if $f = a(t_1, t_2, \ldots, t_n)$ is an update base fact,

$a(t_1, t_2, \ldots, t_n, < f, Q >) \in \Delta EDB^p$, where $Q(f) = c(f)$.

The proof construction algorithm follows.

Proof Construction Algorithm
Input: $DB^p = EDB^p \cup IDB^p$: a logic database augmented proofs;
Output: Y^p: set of proofs;
Method:

```
Begin
$\Delta Y^p := EDB^p$;
$Y^p := \emptyset$;
while $\Delta Y^p \neq \emptyset$ do
    $\Delta Y^p_{pivot} := diff\_I^p(Y^p, \Delta Y^p)$;
    $Y^p := Y^p \cup \Delta Y^p$;
    $\Delta Y^p := \Delta Y^p_{pivot}$;
endwhile;
```

End;

where

$diff\_I^p(Y^p, \Delta Y^p) = \{h(Y, < h(Y)[p_1, p_2, \ldots, p_n], Q >)\theta$:
$h(Y, < h(Y)[p_1, p_2, \ldots, p_n], Q >) \leftarrow a_1(X_1, < p_1, Q_1 >), a_2(X_2, < p_2, Q_2 >$
$), \ldots, a_n(X_n, < p_n, Q_n >) \in IDB^p$,
θ is a substitution such that $\forall i \in [1, n]$, $a_i(X_i, < p_i, Q_i >)\theta \in Y^p \cup \Delta Y^p$ and
$\exists j \in [1, n]$ such that $a_j(X_j, < p_j, Q_j >)\theta \in \Delta Y^p\}$

This algorithm may be called by using the following form:

$$Proof\_Cons(EDB^p, IDB^p, Y^p).$$

We give now the correctness theorem for the proof construction algorithm.

Theorem 2. *Let DB be a logic database; then, by calling*

$$Proof\_Cons(EDB^p, IDB^p, Y^p).$$

we can and can only produce all valid proofs.

To prove this theorem, we introduce the concept of 'proof length'.

Definition 3. Given a proof $< S, Q >$, the length of the longest paths on S from the root to leaves is called the length of this proof.

In fact, our goal is not to explicitly store proofs; instead we are interested in more 'abstract' information, i.e., proof number. Considering proof numbers rather than proof structures migth make us combine and process simultaneously several proofs sharing a common portion. To this end, we introduce a new set operation, denoted as $\uplus$, intended to accumulate proof numbers for every theorems. The $\uplus$ operation is an ordinary $\cup$ operation followed by a rectification, namely, combining all tuples with the same theorem contents into a single one by adding their counters. $DB^c = EDB^c \cup IDB^c$ is the logic database augmented with counters. That is,

- If $f = a(t_1, t_2, \ldots, t_n)$ is a base fact, then, $a(t_1, t_2, \ldots, t_n, c(f)) \in EDB^c$
- If $h(Y) \leftarrow a_1(X_1), a_2(X_2), \ldots, a_n(X_n) \in IDB$ where Y, X_i are vector variables and $n > 0$, then,
$h(Y, c_1 \times c_2 \times \ldots \times c_n) \leftarrow a_1(X_1, c_1), a_2(X_2, c_2), \ldots, a_n(X_n, c_n) \in IDB^c$.

Similarly to EDB, ΔEDB may be augmented with counters to yield ΔEDB^c. If $f = a(t_1, t_2, \ldots, t_n)$ is an update base fact,

$a(t_1, t_2, \ldots, t_n, c(f)) \in \Delta EDB^c$.

Replacing the $\cup$ operation with the $\uplus$ operation, and without considering proof trees, we obtain the proof counting algorithm.

Proof Counting Algorithm
Input: $DB^c = EDB^c \cup IDB^c$: a logic database augmented with counters;
Output: Y^c: set of theorems;
Method:

```
Begin
$\Delta Y^c := EDB^c$;
$Y^c := \emptyset$;
while $\Delta Y^c \neq \emptyset$ do
    $\Delta Y^c_{pivot} := diff\_I^c(Y^c, \Delta Y^c)$;
    $Y^c := Y^c \uplus \Delta Y^c$;
    $\Delta Y^c := \Delta Y^c_{pivot}$;
endwhile;
End;
```

where,

$diff\_I^c(Y^c, \Delta Y^c) = |\{\{h(Y, c_1 \times c_2 \times \ldots \times c_n)\theta :$
$h(Y, c_1 \times c_2 \times \ldots \times c_n) \leftarrow a_1(X_1, c_1), a_2(X_2, c_2), \ldots, a_n(X_n, c_n) \in IDB^c, \theta$ is a substitution such that $\forall i \in [1, n], a_i(X_i, c_i)\theta \in Y^c \cup \Delta Y^c$ and $\exists j \in [1, n]$ such that $a_j(X_j, c_j)\theta \in \Delta Y^c\}\}|$

$\{\{\}\}$ denotes multi-set, where duplicates are not eliminated, and $||$ is the rectifying operation, merging elements with the same theorem contents into a single one by adding their counters.

The algorithm can be called by using,

$$Proof\_Count(EDB^c, IDB^c, Y^c).$$

To prove the correctness of the algorithm, one should verify that for any theorem, the value of the counter attribute yielded by 'Proof Counting Algorithm' is equal to the number of proofs produced by 'Proof Construction Algorithm'. With the concept below, which will be found helpful for further discussions, this correctness may be expressed in a more general manner.

Definition 4. For a given proof $< S, Q >$, the product of integers associated to each of its leaves is called the weight of this proof.

It's clear that the weight of a valid proof is equal to 1.

Theorem 5. *Let DB be a logic database; then, for any theorem t, the value of the counter attribute of t yielded by calling*

$$Proof\_Count(EDB^c, IDB^c, Y^c).$$

is equal to the sum of the weights of all the proofs of t yielded by calling,

$$Proof\_Cons(EDB^p, IDB^p, Y^p).$$

Tuples with the counter value equal to 0 are deleted automatically during and after the computation in the proof counting algorithm.

4 Theorem Maintenance

In this section, we study the following problem. Given a logic database DB, and the set of theorems derived from it. If the EDB receives an update, how to obtain the new set of theorems without obviously recomputing from scratch? Like the last section, we begin our discussion on more concrete information: proof structures; we then shift to proof numbers.

Recall that situations considered are ordinary in that deleted facts are included in the initial axiom base, and the set of inserted facts is disjoint with the initial axiom base.

4.1 Maintenance: Proof Tree

We are interested in proof tree constructions and in comparing the sums of proof weights generated by both proof construction and maintenance algorithms, it is then to be expected that these two algorithms will produce the same set of proof structures. To this end, we will take inserted EDB as a part of inputs in the proof construction algorithm, these inserted elements having counters equal to 0. Suppose that the old set of proofs is produced by the call:

$$\text{c1)}\ Proof\_Cons(EDB^p \cup (\Delta EDB_0^+)^p, IDB^p, Y^p).$$

c1 obviously produces the same set of valid proofs as the call:

$$\text{c2)}\ Proof\_Cons(EDB^p, IDB^p, Y^p).$$

When the EDB is updated, if we derive the new set of proofs from scratch, we can do it by the call:

$$\text{c3)}\ Proof\_Cons((EDB \cup \Delta EDB^+ - \Delta EDB^-)^p \cup (\Delta EDB_0^-)^p, IDB^p, Y^p).$$

Recomputing from scratch in general is not an efficient method, because updates might possibly influence a small portion of the set of derived theorems, thus only this part should be reconsidered. To take advantage of already available information, we propose an incremental computation, which is also based on the semi-naive evaluation. So, it is quite easy to implement.

First, let us revisit the semi-naive evaluation (c.f., Section 2). The major part of this algorithm is a 'while' loop, each iteration corresponds to an application of the recursive derivation rule, which may contribute to 'Δ'; the loop does not terminate until Δ is empty.

The semi-naive evaluation is set-oriented, that is, the order in which facts appear in the axiom base must not affect the final results of the derivation. While dealing with updates, inserted/deleted facts can be regarded as initially present but inactivated axioms: therefore when derivation is performed only using active data, we get the initial set of theorems. When update occurs and maintenance is required, one should only re-activate initially 'dumb' data, and continue with the derivation process.

The proof maintenance algorithm follows.

Proof Maintenance Algorithm
Input: $DB^p = EDB^p \cup IDB^p$: a logic database augmented with proofs;
Y^p: old set of proofs (produced by call c1); ΔEDB^p: update to EDB^p
Output: Y^p: new set of proofs;
Method:

```
Begin
1) $\Delta Y^p := \Delta EDB^p$;
2) while $\Delta Y^p \neq \emptyset$ do
3)     $\Delta Y^p_{pivot} := diff\_I^p(Y^p, \Delta Y^p)$;
4)     $Y^p := Y^p \cup \Delta Y^p$;
5)     $\Delta Y^p := \Delta Y^p_{pivot}$;
6) endwhile;
End;
```

The algorithm may be called by c4:

$$\text{c4) } Maintain\_Proof(Y^p_{old}, EDB^p, IDB^p, \Delta EDB^p, Y^p_{new}).$$

Recall that a proof is characterized by both its structure and the mapping from the set of leaves to $\{-1, 0, 1\}$. Therefore, given a proof structure, one may construct different proofs. If S is a proof structure with w leaves $f_1, f_2, \ldots, f_w$, we use S$(\alpha_1, \alpha_2, \ldots, \alpha_w)$ to denote the proof where the w leaves are mapped respectively to w integers $\alpha_1, \alpha_2, \ldots, \alpha_w$. Among the w leaves, v leaves, say for example $f_1, f_2, \ldots, f_v$ are to be updated. If f_i is updated, $\Delta\alpha_i$ represents the f_i's counter value in ΔEDB^c. It is clear $\Delta\alpha_i$ is equal to 1 or -1.

It should be noted that considering proof numbers instead of proof structures permits to process several different proof structures simultaneously. This may substantially improve performance.

In general, we have the following theorem.

Theorem 6. *Suppose S is a proof structure with w leaves $f_1, f_2, \ldots, f_w$, the proof $S(\alpha_1, \alpha_2, \ldots, \alpha_w)$ is produced by call c1 (see the beginning of 4.1); if $f_1, f_2, \ldots, f_v$ correspond to v updated base facts, then any proof of the form :*

$$S(\underline{\alpha_1}, \underline{\alpha_2}, \ldots, \underline{\alpha_v}, \alpha_{v+1}, \ldots, \alpha_w) \text{ where } \underline{\alpha_i} = \alpha_i \text{ or } \Delta\alpha_i,\ 1 \leq i \leq v,$$

can be produced by call c4:

From this theorem one can easily conclude:

Corollary 7. *Given a proof structure S, the weight of the proof of structure S produced by call c3 is equal to the sum of the weights of all proofs of structure S produced by call c4.*

4.2 Maintenance: Theorem Counter

In the present, we discuss our main algorithm: counter maintenance algorithm. The essential idea of this algorithm is the same as that of the proof maintenance algorithm. They are different only in their manipulations. Like the proof counting algorithm, the counter maintenance algorithm deals only with the proof number. Therefore, several different proofs sharing a common segment might be treated jointly if $\cup$ operation is replaced by $\uplus$ operation.

The counter maintenance algorithm follows.

Counter Maintenance Algorithm
Input: $DB^c = EDB^c \cup IDB^c$: a logic database augmented with counters;
Y^c: old set of theorems; ΔEDB^c: update to EDB^c
Output: Y^c: new set of theorems;
Method:

```
Begin
$\Delta Y^c := \Delta EDB^c$;
while $\Delta Y^c \neq \emptyset$ do
    $\Delta Y^c_{pivot} := diff\_I^c(Y^c, \Delta Y^c)$;
    $Y^c := Y^c \uplus \Delta Y^c$;
    $\Delta Y^c := \Delta Y^c_{pivot}$;
endwhile;
End;
```

The algorithm may be called by:

c5) $Maintain\_Count(Y^c_{old}, EDB^c, IDB^c, \Delta EDB^c, Y^c_{new})$.

Theorem 8. *If t is any theorem, then the counter value of t produced by the counter maintenance algorithm, i.e., call c5, is equal to the sum of the weights of all the proofs produced by call c4 (c.f., 5.1), of which the root is t.*

Corollary 9. *The counter maintenance algorithm correctly maintains the set of theorems.*

Finally, it should be noted that in the counter maintenance algorithm, tuples with the counter value equal to 0 are deleted automatically during and after the computation.

5 Conclusion

We proposed and investigated a method for efficiently and incrementally maintaining recrusively defined data. The main contributions are two algorithms: proof counting algorithm and counter maintenance algorithm. We suggest to use the first algorithm for derivation and second for maintenance. The approach is based upon the semi-naive evaluation technique, it is therefore quite easy to implement. Multiple updates of different natures (insertion or deletion) are handled uniformly, and within a single processing. In order to get a concrete idea about the performance of our proposed method, a system was prototyped on Sparc stations. Experimental evaluations shown that it is efficient when the underlying data is not 'too' connected, or updated base facts are not too 'numerous'.

References

1. F. BRY : 'Query Evaluation in Recursive Databases: Bottom-up and Top-Down Reconciled', *Data and Knowledge Engineering*, 5, 1990, 289-312.
2. F. BANCILHON, D. MAIER, Y. SAGIV, J.D. ULLMAN : 'Magic Sets and Other Strange Way to Implement Logic Programs' , In *Proceedings ACM PODS*, 1986.
3. F. BANCILHON, R. RAMAKRISHNAN : 'An Amateur's Introduction To Recursive Query Processing Strategies' , In *Proceedings ACM SIGMOD 1986 International Conference on Management of Data*, Washington, USA, May 1986.
4. J.-P. CHEINEY, Y.-N. HUANG: 'Set-orinted Propagation of Updates into Transitively Closed Relations', In *Proceedings 2nd Int'l Conf. on DOOD*, Munich, Germany, Dec. 1991.
5. J.-P. CHEINEY, Y.-N. HUANG: 'Efficient Maintenance of Explicit Transitive Closures with Set-oriented Update Propagation and Parallel Processing', to appear in *Data and Knowledge Engineering.*
6. J.W. FLOYD : 'Foundations of Logic Programming', Spring-Verlag Germany, 1987.
7. A. GUPTA, I.S. MUMICK, V.S. SUBRAHMANIAN : 'Maintaining Views Incrementally', In *Proceedings ACM SIGMOD 1993 International Conference on Management of Data*, Washington, DC, USA, May 1993.
8. J. HAN, L. LIU : 'Efficient Evaluation of Multiple Linear Recursions', In *T.K.D.E*, VOL. 17, NO. 12, Dec. 1991.
9. Y.N. HUANG, J.P. CHEINEY :'Differential Maintenance of a Transitively Closed Relation', In *Proceedings 5th International Symposium on Computer and Information Sciences*, Nevsehir Cappadocia, Turkey, November 1990.
10. Y.N. HUANG, V. DAHL, J. HAN :'Fact Updates in Logic Databases', Technical report of Simon Fraser University, CSS/LCCR TR94-06, 1994.
11. J.D. ULLMAN: 'Principles of Database and Knowledge-Base Systems', Vol1 and Vol2, Computer Science Press, New York, USA, 1988.
12. L. VIEILLE: 'Recursive Axioms in Deductive Databases: the Query sub-Query Approach', In *Proceeding 1st Int'l Conf. on Expert Database Systems*, Charleston, South Carolina, USA, 1986.

On the Relationship between Assumption-based Framework and Autoepistemic Logic

Y.J. Jiang* and Yongyuth Aramkulchai**
Department of Computing, Imperial College
London SW7 2BZ, England
{yj,ja8}@uk.ac.ic.doc

Abstract. Assumption-based (AB) framework [BTK93] is a generalisation of abduction. It augments a theory with a set of *admissible* assumptions that it can defend against any attacks. Autoepistemic logic is a nonmonotonic logic about an ideal agent who reasons about her beliefs and ignorance *introspectively.* On the surface, these two formalisms look very different. A closer examination reveals many similarities. Indeed, BTK93 has shown that *stable extensions* in the AB framework of an autoepistemic theory correspond to the Moore's *AE extensions* of the theory. In this paper, we shall continue to establish some more relationships between the two formalisms. We first reformulate BTK's AB framework for AE logic with a *modal* approach. We then provide an AB framework for a revised *preferred extension* that neither subsumes nor being subsumed by Przymusinski's 3-valued AE logic. We then show that the AB framework for a revised *complete extension* of any *consistent* AE theory strictly subsumes Przymusinski's 3-valued AE logic. By exploiting the clear structure of the modally reformulated AB framework, we further define an AB framework for a reflexive *preferred* (cf. *stable*, *complete*) *extension* that integrates *Schwarz's reflexive AE logic* with Przymusinski's 3-valued AE logic.

Keywords Assumption-based framework, Autoepistemic Logic, Argumentation system, Reflexive Autoepistemic Logic, Abduction, 3-valued AE logic

1 Introduction

All nonmonotonic formalisms can be regarded as consisting of two components: *assumptions* and their *preference* relationships in the presence of incompatible assumptions. For example, in Reiter's closed world formalisation of a database [Ri78], assumptions are literals that the database does not entail and preference is to choose such literals to be negative atoms. This observation essentially forms the foundation of assumption-based (AB) framework [BTK93] as a generic model of nonmonotonic reasoning. The AB framework characterizes preference relationships among assumptions based on Dung's [Du93] notion of *admissibility* which involves *attack* and *defence* between two sets of assumptions. It extends a theory with a set of *admissible* assumptions that the set can defend

* Advanced fellow of British Science and Engineering Research Council.
** Supported financially by the Royal Thai Government Scholarship.

against any attacks. Depending on the properties of admissible assumptions, different specific formalisms of the AB framework can be derived. *Stable* extension is defined on the basis of the set of admissible assumptions that attacks everything outside the set. Whilst *preferred extension* is defined on the basis of the *maximum* set of admissible assumptions that the set can defend.

Autoepistemic (AE) logic is a nonmonotonic logic about an ideal agent who reasons about her beliefs and ignorance by introspection. It forms a closure called *AE extension* for an AE theory by *negative introspection* or *positive introspection* or neither. Negative introspection is to include non-belief of a formula to the closure if the formula is not in the closure and positive introspection is to include belief of a formula to the closure if the formula is in the closure. If neither introspection is used for a formula, a 3-valued AE logic such as Przymusinski's [Pr91] can be induced. Nonmonotonicity is accommodated in AE logic by the property that a conclusion based on what is currently believed or not believed can be withdrawn in the presence of new beliefs. Like the AB framework, there are many forms of autoepistemic logic. These forms mainly differ in their construction of *AE extension* of an AE theory. For example, McDermott and Doyle's AE logic [MD80] is closed under *negative introspection*; while Moore's AE logic [Mo85] is *additionally* closed under *positive introspection* which is further modified in Schwarz's *reflexive* AE logic [Sc91]. In fact, Schwarz has shown that all these variations of AE logic can be captured by McDermott and Doyle's general AE logical framework where *a modal system* is used to close the AE theory under negative introspection. For example, the modal system in Moore's AE logic will be **KD45**; while in Schwarz's reflexive AE logic, it will be **SW5**.

On the surface, AE and AB look very different. A closer examination reveals many similarities. Fundamentally, positive introspection and negative introspection in AE can be regarded as assumptions in the AB framework. The preference relationship will simply be to favour assuming non-belief of a formula more than assuming the belief of the formula unless the formula is actually believed. Indeed, this is precisely what BTK93 has observed. They have shown that *stable extensions* in the AB framework of an AE theory is equivalent to Moore's AE extensions [Mo85] of the theory. The purpose of this paper is to carry the BTK's work further. It is envisaged that by studying more on the relationships between the AB framework and AE logic, we can both demonstrate the generality of the AB framework and discover enhancements to the framework from the insights of the various forms of AE logic. In particular, by reformulating various forms of AE logic within the AB framework, we can provide a syntactical characterization for each form that may not be present before. Conversely, since AE has a clear semantics [MT91], it may be regarded to provide a semantical characterization of the AB framework.

Specifically, the contributions of this paper are

1. the revision of BTK's AB framework using a modal approach.
2. the establishment of the relationship between revised preferred (cf. complete) extensions and Przymusinski's 3-valued AE extensions of an AE theory; in particular, the provision of a syntactical characterization of Przymusinski's 3-valued AE logic of any consistent AE theory.

3. the natural transformation of the modally formulated AB framework to Schwarz's reflexive AE logic.
4. the introduction of a *reflexive preferred* (cf *complete*) *extension* that integrates Schwarz's reflexive AE logic with Przymusinski's 3-valued AE logic.

The basic idea of our approach is quite simple. It uses the modal formulation of various forms of AE logic to define consistency, attack and admissibility notions in the AB framework. In this way, it is very natural to establish the relationship between the AB framework and AE logic. The advantage of the modal formulation is that the concept of stable extension, preferred extension and complete extension can be straightforwardly transfered to *reflexive stable extension*, *reflexive preferred extension* and *reflexive complete extension.*

This paper is organized as follows. We introduce the AB framework for AE logic and the AE logic in Section 2 and 3, respectively. In Section 4, we present a modal reformulation of BTK's AB framework for AE logic. In particular, we establish the relationship between preferred (cf. complete) extension and Przymusinski's 3-valued AE logic. In Section 5, we present the reflexive stable extension that corresponds to schwarz's reflexive AE logic and the reflexive preferred (cf complete) extension that improves on Schwarz's reflexive AE logic and Przymusinski's 3-valued AE logic.

2 The AB Framework for AE Logic

To relate the AB framework with AE logic, BTK first define the following instantiations of a general AB framework [BTK93].

Definition 1 *The AB framework for an AE theory is* $\langle\langle\mathcal{L},\mathcal{R}\rangle,Ab,\leq\rangle$

1. $\mathcal{L}$ *consists of the full AE language*
2. $\mathcal{R}$ *is the ordinary propositional system Th augmented with an extra rule of inference "from* $\{\phi, \neg\mathbf{L}\phi\}$*, infer* $\perp$*". We denote the resultant system by Th'.*
3. *Ab is* $\{\neg\mathbf{L}\phi \mid \phi \in \mathcal{L}\} \cup \{\mathbf{L}\phi \mid \phi \in \mathcal{L}\}$.
4. $\leq$ *is defined by* $\phi \leq \neg\mathbf{L}\phi$ *and* $\neg\mathbf{L}\phi \leq \mathbf{L}\phi$*, for all* $\phi \in \mathcal{L}$.

The preference relationship is used to define the attack in Dung's augmentation framework.

Definition 2 *Let* $\vdash$ *denote the derivability relationship of Th. A set of assumptions A* **attacks** *a set of assumptions* Δ *with respect to an AE theory T if and only if it satisfies one of the following conditions.*

1. $T \cup A \vdash \phi$ *and* $\neg\mathbf{L}\phi \in \Delta$
2. $T \cup A \vdash \neg\mathbf{L}\phi$ *and* $\mathbf{L}\phi \in \Delta$

To ensure that only "sensible assumptions" are added to a theory, we define

Definition 3 *A set of assumptions* Δ *is* **admissible** *with respect to an AE theory T if and only if it satisfies the following conditions.*

1. $T \cup \Delta \not\vdash_{\mathcal{R}} \bot$, *ie. the added assumptions should be consistent with the theory.*
2. *for all set of assumptions* $A \in Ab$, *if* A *attacks* Δ, *then* Δ *attacks* A.

Ideally, we would like to extend a theory by all the assumptions that can be defended.

Definition 4 *A set of assumptions* Δ *is* stable *with respect to an AE theory* T *if and only if it is admissible and attacks* $\{\phi\}$ *for any* $\phi \in Ab$ *and* $\phi \notin \Delta$.

Unfortunately, there may not exist a stable set of assumptions for a theory in which case we would like to add as many admissible assumptions as possible.

Definition 5 *A set of assumptions* Δ *is* preferred *with respect to an AE theory if and only if* Δ *is maximally (with respect to set inclusion) admissible.*

Or even weaker, we may just want to construct a set of assumptions that contains everything the set can defend.

Definition 6 *A set of assumptions* Δ *is* complete *with respect to an AE theory if and only if*

1. $T \cup \Delta \not\vdash_{\mathcal{R}} \bot$
2. $\Delta = \{\alpha \mid \alpha \in Ab \wedge (\forall A \subseteq Ab, A \text{ attacks } \{\alpha\} \rightarrow \Delta \text{ attacks } A)\}$

Dung and BTK have shown that

Theorem 1 *A set of stable assumptions is also preferred; but not the other way around. A set of preferred assumptions is also complete; but not the other way around.*

Corresponding to these different kinds of assumptions added to an AE theory, we define their closures with the theory as follows.

Definition 7 *E is a* stable *(cf.* preferred, complete*)* extension *of an AE theory T if and only if there exists a* stable*} (cf.* preferred, complete*) set of assumptions* Δ *such that*

$$E = \{\phi \mid T \cup \Delta \vdash_{\mathcal{R}} \phi\}$$

Clearly, we have

Theorem 2 *A preferred extension of an AE theory is a complete extension of the theory. And there always exists a preferred (hence complete) extension for any consistent theory.*

Theorem 3 *E is an AE extension of an AE theory if and only if it is a stable extension of the AE theory in the AB framework.*

Consider the AE theory {**L**p → p}. Moore's AE logic yields two AE extensions whose objective parts are Cn({p}) and Cn({}) and the AB framework also provides those two stable extensions.

The BTK's AB framework however does not seem to be naturally expandable to preferred extension. Under their framework, given the AE theory {**LL**p} which has no stable extension, the preferred extension will include $\{\neg \mathbf{L}p\}$ as admissible. The BTK framework also did not establish any relationship between the preferred extension of the AB framework for an AE theory and any existing variation of AE logic. Nor it attempts to relate to the reflexive AE logic.

3 Autoepistemic Logics

The language $\mathcal{L}$ of AE is an *ordinary* propositional logic augmented with a modal operator **L**. The *basic semantics* of AE is characterized by a list-like valuation (I,T) where I is a set of ordinary propositional letters and T a set of AE formula. The satisfiability relationship between a valuation (I,T) and an AE formula is defined in a standard way except for ordinary atoms and modal literals. For an ordinary atom p, $\models_{(I,T)} p$ iff $p \in I$; whilst for a modal literal $\mathbf{L}\phi$, $\models_{(I,T)} \mathbf{L}\phi$ iff $\phi \in T$.

Definition 8 *We define an AE theory to be a set of AE formula. We define the objective part of an AE theory to be the set of ordinary formulae in the theory. We use Cn to denote the ordinary propositional closure.*

The basic idea of AE logic is to form an ideal closure called *extension* over an AE theory. This notion is captured at *meta-level* [Le90] through a fixpoint construction that involves the *basic semantics* of AE. Moore has shown that this construction is equivalent to the following syntactical fixpoint definition[1].

Definition 9 *E is an AE extension of an AE theory A iff it satisfies the following equation:*

$$E = Cn(A \cup \{\mathbf{L}\alpha \mid \alpha \in E\} \cup \{\neg \mathbf{L}\beta \mid \beta \notin E\})$$

This fixpoint definition however does not guarantee the existence of AE extension for *any* AE theory. For example, the AE theory {**L**p} has no coherent AE extension. Another example is the AE theory $\{\neg \mathbf{L}p \rightarrow p$, r} because **L**p is somewhat paradoxical. The incoherence of the first example seems against commonsense since **L**p itself is a belief after all. The incoherence of the second example seems also against commonsense reasoning as r after all is a sensible belief irrespective of what $\neg \mathbf{L}p \rightarrow p$ means. These problems prompt the development of Schwarz's *reflexive AE logic* and Przymusinski's *3-valued AE logic.*

Schwarz's reflexive logic [Sc91] treats beliefs of an agent as knowledge from the agent point of view. In this logic, a *reflexive AE extension* is defined as follows.

[1] Throughout this paper, we shall always use a syntactic definition of an extension if possible.

Definition 10 *E is a reflexive AE extension of an AE theory A iff it satisfies the following equation:*

$$E = Cn(A \cup \{\mathbf{L}\alpha \equiv \alpha \mid \alpha \in E\} \cup \{\neg\mathbf{L}\beta \mid \beta \notin E\})$$

Thus given the AE theory $\{\mathbf{L}\mathrm{p}\}$, the reflexive AE logic can ground $\mathbf{L}\mathrm{p}$ with the assumption of $\mathbf{L}p \equiv p$. And given the AE theory $\{\neg\mathbf{L}p \rightarrow p, \mathrm{r}\}$, the theory is coherent in the reflexive AE logic because p is true on the assumption of $\mathbf{L}p \equiv p$. Schwarz justifies the reflexive behaviour of $\neg\mathbf{L}p \rightarrow p$ as "even if not p, still p".

Despite of these improvements, there are still some question marks over the reflexive AE logic. First, given $\{\mathbf{L}p \rightarrow p\}$, there is *only one* reflexive AE extension which contains $\neg\mathbf{L}p$ instead of *two* AE extensions one of which also contains $\mathbf{L}\mathrm{p}$ in Moore's AE logic. This position is of course double-edged. Schwarz justifies his position by arguing that a belief should not be grounded on the assumption that it is believed. While this viewpoint may be sensible in some applications, Moore's multiple AE extensions are also useful in logic programming, especially when we try to handle positive loops such as $\{p : -p\}$ by formalizing it as $\{\mathbf{L}p \rightarrow p\}$ [Ji93b, Bo92]. Furthermore, Schwarz's argument is not entirely justified when we consider the example $\{\mathbf{L}p \rightarrow q,\ \mathbf{L}(p \rightarrow q) \rightarrow p\}$. In addition to the extension containing the empty objective part Cn({}), this example also produces another reflexive AE extension containing p and q which are justified by the assumptions that $\mathbf{L}p \equiv p$, $\mathbf{L}q \equiv q$ and $\mathbf{L}(p \rightarrow q) \equiv (p \rightarrow q)$.

It is also felt that Schwarz's logic may produce some unintuitive extensions. Consider for example, the AE theory $\{\neg\mathbf{L}p \wedge \neg\mathbf{L}q \rightarrow q,\ \neg\mathbf{L}q \rightarrow p\}$ which has a *unique* AE extension containing p. However the theory has an *additional* reflexive AE extension containing q. This extension does not seem to be intuitive since the meaning of the first formula in the theory which is often intended as an cancellation axiom in commonsense reasoning [Ge88] is not captured. Perhaps most important of all, reflexive AE logic still cannot yield a fixpoint extension for some AE theories such as $\{\mathbf{L}\neg\mathbf{L}p \rightarrow p\}$ although intuitively this should behave identically as $\{\neg\mathbf{L}p \rightarrow p\}$ as it is the case in Moore's AE logic. This example however can be dealt with in Przymusinski's 3-valued AE logic [Pr91].

The language of the 3-valued AE logic is identical to that of AE logic except that every $\mathbf{L}\phi$ is considered as a propositional atom. The possible truth values for each atom are {1,0,1/2} which corresponds to {true,false,undefined} respectively.

Definition 11 *A 3-valued interpretation of 3-valued AE logic is a mapping from every propositional atom to a unique truth value of {1,0,1/2}.*

Definition 12 *The 3-valued truth valuation $\hat{I}$ of an AE formula based on a 3-valued interpretation I is defined as follows.*

$$\begin{aligned}
&\hat{I}(A) = I(A) \textit{ where } A \textit{ is a propositional atom}\\
&\hat{I}(\neg A) = 1 - \hat{I}(A)\\
&\hat{I}(A \wedge B) = min(\{\hat{I}(A), \hat{I}(B)\})\\
&\hat{I}(A \vee B) = max(\{\hat{I}(A), \hat{I}(B)\})\\
&\hat{I}(A \rightarrow B) = \begin{cases} 1 \textit{ if } \hat{I}(B) \geq \hat{I}(A) \\ 0 \textit{ otherwise} \end{cases}
\end{aligned}$$

Definition 13 *A 3-valued interpretation I is a 3-valued model of an AE theory T if* $\forall\phi \in T$, $\hat{I}(\phi) = 1$.

Definition 14 *A 3-valued belief interpretation M is a mapping from every propositional atom of the form* $\mathbf{L}\phi$ *to a unique truth value of {1,0,1/2}.*

Definition 15 *Given a 3-valued belief interpretation M, let* $Mod3_T(M)$ *denote the set of all 3-valued models of an AE theory T, in which the interpretation of atoms of the form* $\mathbf{L}\phi$ *is given by M.*

Definition 16 *A 3-valued AE extension E of an AE theory T is defined as*

$$E = Cn(T \cup \{\mathbf{L}\phi \mid M(\mathbf{L}\phi) = 1\} \cup \{\neg\mathbf{L}\phi \mid M(\mathbf{L}\phi) = 0\}$$

where M is a 3-valued belief interpretation which satisfies the following condition:

$$M(\mathbf{L}S) = Val3_M{}^T(S)$$

where $Val3_M{}^T(S) = min(\{\hat{m}(S) \mid m \in Mod3_T(M))\})$

For example, the AE theory $\{\mathbf{L}\neg\mathbf{L}p \rightarrow p\}$ has a 3-valued AE extension in which p will be undefined, ie. neither $\mathbf{L}$p nor $\neg\mathbf{L}p$ (hence neither $\mathbf{L}\neg\mathbf{L}p$) is in the extension. However the 3-valued AE logic is still incoherent for some AE theories such as {$\mathbf{L}$p} which can be dealt with in the reflexive AE logic.

In fact, both AE logic and reflexive AE logic can be seen as special cases of McDermott and Doyle's general autoepistemic logical framework [MD80, McD82] in the following fixpoint definition.

Definition 17 *Let A be an AE theory and* $\mathbf{S}$ *be any monotonic modal logic on the modal operator* $\mathbf{L}$. *Then E is a* $\mathbf{S}$*-extension of A iff*

$$E = Cn_{\mathbf{S}}(A \cup \{\neg\mathbf{L}\phi \mid \phi \notin E\})$$

where $Cn_{\mathbf{S}}$ *denote the consequence closure operator of the* $\mathbf{S}$ *modal logic.*

Konolige has shown [Ko88] that

Theorem 4 *Let* **KD45**[2] *be the modal logic obtained from the* **S5** *modal logic by replacing the axiom schema* **T** $(\mathbf{L}\phi \rightarrow \phi)$ *by the weaker schema* **D** $(\mathbf{L}\phi \rightarrow \neg\mathbf{L}\neg\phi)$. *Then E is a Moore's AE extension of A iff E is a* **KD45***-extension of A.*

Schwarz has shown [Sc91] that

Theorem 5 *Let* **SW5** *be the modal logic obtained from* **S5** *by replacing the* 5*-axiom schema* $(\neg\mathbf{L}\phi \rightarrow \mathbf{L}\neg\mathbf{L}\phi)$ *by the weaker schema* $(\phi \rightarrow (\neg\mathbf{L}\phi \rightarrow \mathbf{L}\neg\mathbf{L}\phi))$. *Then E is a reflexive AE extension of A iff E is a* **SW5***-extension of A.*

[2] We assume the reader is familiar with various axiomatic systems in modal logic. The 4 axiom is $\mathbf{L}\phi \rightarrow \mathbf{LL}\phi$.

Clearly, given any modal logics **S** and **U**, an **S**-extension is a **U**-extension if **S** is a modal logic contained by **U**. Schwarz [Sc91] has also shown that $S \in \{\mathbf{KD45}, \mathbf{SW5}\}$ are the maximal **S**-extensions that are not collapsed to a monotonic logic as it is the case for **S5**-extension. This means that instead of using **KD45**- or **SW5**- extensions, we can also use any other weaker extensions such as **T**-extensions based on the modal logic T.

However Schwarz did not give any corresponding modal syntactical characterization of the 3-valued AE extension. One of the purposes of this paper is to provide such a syntactical characterization.

4 Revised {Stable, Preferred and Complete} Extensions

4.1 The Revised AB Framework

To extend the BTK's result to other forms of AE logic, we define the following revised AB framework for AE logic. The basic idea is to instantiate $\mathcal{R}$ in the general AB framework by a modal system eg. **KD45**, instead of Th as it is the case in BTK's approach. Also the assumptions need only be negative **L**-literals in the revised AB framework.

Definition 18 *The revised AB framework for Moore's AE logic is* $\langle\langle\mathcal{L},\mathcal{R}\rangle, Ab, \leq\rangle$

1. *$\mathcal{L}$ consists of the full AE language.*
2. *$\mathcal{R}$ is the* **KD45** *propositional system augmented with an extra rule of inference "from $\{\phi, \neg\mathbf{L}\phi\}$, infer $\bot$". We call the resultant inference system* **KD45′**.
3. *Ab is $\{\neg\mathbf{L}\phi \mid \phi \in \mathcal{L}\}$.*
4. *$\leq$ is defined by $\{\phi \leq \neg\mathbf{L}\phi \mid \phi \in \mathcal{L}\}$.*

The revised attack relationship simply replaces $\mathcal{R}$ by **KD45′** in the general AB formulation of attack. The counterattack is defined again as before.

Definition 19 *A set of assumptions Δ* **attacks** *a set of assumptions A with respect to an AE theory T if and only if $T \cup \Delta \vdash_{\mathbf{KD45}'} \phi$ for some $\neg\mathbf{L}\phi \in A$.*

The revised definition of admissible assumptions again simply replaces $\mathcal{R}$ by **KD45'** in the general AB formulation of admissibility.

Definition 20 *A set of assumptions Δ is* **admissible** *with respect to an AE theory T if and only if it satisfies the following conditions.*

1. *$T \cup \Delta \nvdash_{\mathbf{KD45}'} \bot$.*
2. *for all set of assumptions $A \in Ab$, if A attacks Δ, then Δ attacks A.*

The definition of stable assumptions is defined exactly the same as before except it uses the revised attack and admissible definitions. We can now define the revised stable extension as follows by instantiating $\mathcal{R}$ by **KD45′** in the general definition of stable extension.

Definition 21 *E is a* revised **stable extension** *of a theory T if and only if there exists a* **stable** *set of assumptions Δ such that*

$$E = \{\phi \mid T \cup \Delta \vdash_{\mathbf{KD45'}} \phi\}$$

It can be easily shown that

Theorem 6 *E is a* revised **stable extension** *of an AE theory T if and only if it is a Moore's AE extension of the theory.*

For example, given the AE theory $\{\mathbf{L}p \rightarrow p\}$, we can add $\neg\mathbf{L}p$ in a stable set of assumptions because it can defend the attack by $\neg\mathbf{L}\neg\mathbf{L}p$ which implies **L**p (hence p with the AE theory) under the **KD45′** system in the revised AB framework. We can also separately add $\neg\mathbf{L}\neg\mathbf{L}p$ in another stable set of assumptions because it can defend the attack by $\neg\mathbf{L}p$ under the **KD45′** system[3].

4.2 Revised Preferred Extension and 3-valued AE Logic

Similarly, we can naturally extend the revised AB framework to define a revised preferred extension. The definition of the revised preferred assumptions is again defined exactly the same as the preferred assumptions except it uses the revised attack and admissible definitions.

Definition 22 *E is a* revised **preferred extension** *of an AE theory T if and only if there exists a* **preferred** *set of assumptions Δ such that*

$$E = \{\phi \mid T \cup \Delta \vdash_{\mathbf{KD45'}} \phi\}$$

Like preferred extension, we also have

Theorem 7 *There always exists a revised preferred extension for any consistent AE theory*[4].

However the revised preferred extension overcomes some of the problems of nested AE theory that are present in the preferred extension. Consider for example the AE theory {**LLp**} again. Under the ABK framework, we can add $\neg\mathbf{L}p$ because it is not inconsistent with the theory and it is not attacked by anything. On the other hand, the BTK's AB framework cannot add $\neg\mathbf{L}\neg\mathbf{L}p$ because it cannot defend the attack by $\neg\mathbf{L}p$. Both cases are intuitively wrong. In contrast, we cannot add $\neg\mathbf{L}p$ in the revised framework because it is not consistent with the theory under the **KD45′** system. Although, we still cannot add $\neg\mathbf{L}\neg\mathbf{L}p$, however it can be derived by the **KD45′** system in the revised preferred extension of the AE theory.

Unlike the equivalence relationship between stable extension and Moore's AE logic, neither the AB framework of revised preferred extension nor Przymusinski's 3-valued AE logic subsumes the other.

[3] Its defense is possible because $\neg\mathbf{L}\neg\mathbf{L}p$ and the theory together derive p under the **KD45′** system.

[4] Here consistency is defined with respect to the KD45' system instead of an ordinary system as in BTK framework.

Theorem 8 *Neither the set of* revised **preferred extensions** *of an AE theory T is always a strict subset of the set of 3-valued AE extensions; nor is the otherway around.*

Consider for example, $\{\mathbf{L}p\}$, which has no 3-valued AE extension in the 3-valued AE logic. However it has a revised preferred extension $\{\phi \mid \{\mathbf{L}p\} \vdash_{\mathbf{KD45}'} \phi\}$. On the other hand, the AE theory $\{\mathbf{L}p \rightarrow p\}$ has a 3-valued AE extension which has neither $\mathbf{L}p$ nor $\neg\mathbf{L}p$ and is not a revised preferred extension.

Nevertheless, if a consistent AE theory has a 3-valued AE extension, then a revised preferred extension of the theory must be a 3-valued AE extension of the theory. In this case the AB framework for the revised preferred extension can be regarded as less sceptical than 3-valued AE logic although the former is still more sceptical than the well-founded semantics [LJJ91].

Theorem 9 *Given the existence of a Przymusinski's 3-valued AE extension of a consistent AE theory, if E is a* revised **preferred extension** *of a consistent AE theory then it is a Przymusinski's 3-valued AE extension of the theory.*

For example, given the AE theory $\{\neg\mathbf{L}p \rightarrow p\}$, we cannnot add $\neg\mathbf{L}p$. The addition is not admissible to the theory because it is inconsistent under the **KD45**$'$ system. We cannot add $\mathbf{L}p$ either since it is not even an assumption. In addition, we cannot add $\neg\mathbf{L}\neg\mathbf{L}p$ because it cannot defend the attack by $\neg\mathbf{L}p$. These all correspond to 3-valued AE extension of the theory.

Note that the theorem does not hold for any inconsistent[5] AE theory since 3-valued AE logic may yield a 3-valued AE extension while the revised AB framework for preferred extensions does not. Consider an inconsistent AE theory $\{p \leftrightarrow \neg p, (p \leftrightarrow \neg p) \rightarrow q\}$[6] has a 3-valued AE extension containing $p \leftrightarrow \neg p$ and q which does not correspond to any revised preferred extension of the theory. In addition, another inconsistent AE theory under the **KD45'** systems $\{\neg\mathbf{L}p \leftrightarrow \neg\mathbf{L}\neg\mathbf{L}p, \neg\mathbf{L}p \rightarrow p\}$ has a 3-valued AE extension containing $\{\neg\mathbf{L}p \leftrightarrow \neg\mathbf{L}\neg\mathbf{L}p\}$ with a 1/2 value assignment to $\mathbf{L}p$ and $\mathbf{L}\neg\mathbf{L}p$ but there is no revised preferred extension of the theory due to the inconsistency. However, for any consistent AE theory, for instance, $\{\neg\mathbf{L}p \leftrightarrow p\}$ has a 3-valued AE extension which corresponds to a revised preferred extension of the theory.

4.3 Revised Complete Extension and 3-valued AE Logic

Similarly, we can naturally extend the revised AB framework to define a revised complete extension for an AE theory. The definition of the revised complete assumptions is again defined exactly the same as the complete assumptions except it uses the revised attack and admissible definitions involving the **KD45**$'$ modal system.

[5] Note that the inconsistency here refers to that with respect to KD45' which is a 2-valued logic.

[6] Note that the theory is consistent in a 3-valued logic where p's truth value is 1/2 and $p \leftrightarrow \neg p$ can still be 1.

Definition 23 *A set of assumptions Δ is a* revised **complete** *set with respect to an AE theory if and only if*

1. $T \cup \Delta \nvdash_{\mathbf{KD45'}} \bot$
2. $\Delta = \{\alpha \mid \alpha \in Ab \wedge (\forall A \subseteq Ab, A \text{ attacks } \{\alpha\} \rightarrow \Delta \text{ attacks } A)\}$

Definition 24 *E is a* revised **complete extension** *of an AE theory T if and only if there exists a* revised **complete** *set of assumptions Δ such that*

$$E = \{\phi \mid T \cup \Delta \vdash_{\mathbf{KD45'}} \phi\}$$

Like the relationship between preferred extension and complete extension of a theory, we also have

Theorem 10 *A revised preferred extension of an AE theory is also a revised complete extension of the theory; the converse however is not true.*

Theorem 11 *There always exists a revised complete extension for any consistent AE theory.*

Unlike the non-subsumption relationship between the AB framework of revised preferred extension and the 3-valued AE logic, we can show that the AB framework for a revised complete extension strictly subsumes the 3-valued AE logic.

Theorem 12 *If E is a Przymusinski's 3-valued AE extension of a consistent AE theory, then it is also a* revised **complete extension** *of the theory.*

Consider for example, {**Lp**}, which has no 3-valued AE extension in the 3-valued AE logic. However it has a revised complete extension $\{\phi \mid \{\mathbf{L}p\} \vdash_{\mathbf{KD45'}} \phi\}$. Furthermore, the AE theory $\{\mathbf{L}p \rightarrow p\}$ has a 3-valued AE extension with an empty objective part which is also a revised complete extension.

Nevertheless, if a consistent AE theory has a 3-valued AE extension, then the AB framework for a revised complete extension of AE logic coincides with the 3-valued AE logic.

Theorem 13 *Given the existence of a Przymusinski's 3-valued AE extension of a consistent AE theory, E is a* revised **complete extension** *of a consistent AE theory iff it is a Przymusinski's 3-valued AE extension of the theory.*

This theorem can be regarded to provide a syntactical characterization of 3-valued AE extension in the revised AB framework.

5 Reflexive {Stable, Preferred and Complete} Extensions

Although preferred and complete extensions guarantee to exist for any consistent AE theory, they may not be intuitively maximum. Consider for example, the AE theory {**Lp**, $p \rightarrow q$}. This theory has a preferred extension containing

Lp, but does not contain p nor q both of which should intuitively be derived. Consider another example, $\{\neg \mathbf{L} abnormal \rightarrow q\}$ which could be used in fault diagnosis to mean something like "if it is not believed to be abnormal, then q". Suppose we observe that $\neg q$, then we should conclude the component is abnormal. However preferred extension only concludes that it is believed to be abnormal. These problems however can be solved in the reflexive AE logic.

One major advantage of the revised AB framework for AE logic is that it can be straightforwardly transfered into an AB framework for reflexive AE logic.

Definition 25 *The reflexive AB framework for reflexive AE logic is* $\langle\langle\mathcal{L},\mathcal{R}\rangle, Ab, \leq\rangle$

1. *$\mathcal{L}$ consists of the full AE language.*
2. *$\mathcal{R}$ is the* **SW5** *propositional system augmented with an extra rule of inference "from $\{\phi, \neg\mathbf{L}\phi\}$, infer $\perp$". We call the resultant inference system* **SW5′**.
3. *Ab is $\{\neg\mathbf{L}\phi \mid \phi \in \mathcal{L}\ \}$.*
4. *$\leq$ is defined by $\{\phi \leq \neg\mathbf{L}\phi \mid \phi \in \mathcal{L}\}$.*

The attack relationship simply replaces $\mathcal{R}$ by **SW5′** in BTK's AB formulation of attack for AE logic and other relevant definitions are also defined in the same way. Due to space limitation, we will not give here the definitions which can be seen easily.

Theorem 14 *E is a* reflexive **stable extension** *of an AE theory T if and only if it is a Schwarz's reflexive AE extension of the theory.*

Theorem 15 *There always exists a reflexive preferred or complete extension for any consistent AE theory.*

The reflexive complete extension can be seen as a combination of the virtues of reflexive AE logic and the 3-valued AE logic. Unlike both logics, there is always a reflexive complete extension for any AE theory. In particular, a reflexive complete extension exists for the AE theory $\{\mathrm{L}p, p \rightarrow q\}$ which has no 3-valued AE extension. And this extension can derive p and q both of which are not derivable in the non-revised complete extension. The AB framework for a reflexive preferred extension can be seen as somewhere intermediate between the 3-valued AE logic and the AB framework for a reflexive complete extension.

Conclusion

In this paper, we have presented a reformulation of the AB framework for AE logic using a modal approach. Using the reformulated framework under the modal system KD45, we have shown that stable extensions of an AE theory correspond to Moore's AE extensions of the theory. We have also shown that revised preferred extensions of an AE theory and the 3-valued AE extensions of the theory do not subsume the other. On the other hand, we have shown that the AB framework for a revised complete extension of any consistent AE theory subsumes the 3-valued AE logic. Particularly, we have indicated that the

AB framework for a revised complete extension of any consistent AE theory provides a syntactical characterization of the 3-valued AE logic.

By instantiating the modally formulated AB framework with the modal logic **SW5**, we have also developed a notion of reflexive stable extension that corresponds to the notion of reflexive AE extension in Schwarz's reflexive AE logic. By naturally extending this reflexive AB framework to maximum admissible assumptions and complete assumptions, we have also obtained a notion of reflexive preferred or complete extension that combines the virtues of reflexive AE logic and the 3-valued AE logic. We have shown that there always exists a reflexive preferred or complete extension for any consistent AE theory.

The modal-based AB framework for AE logic can be generalized to other forms of AE logic by simply changing the modal system, eg. to **T**. In particular, we can naturally extend the new framework to deal with multiagent nonmonotonic reasoning by substituting a monotononic epistemic logic into the framework in the spirit of [Ji94]. Despite of the nature of argumentative system upon which the AB framework is partly based, it is still restricted to a single agent. The modal formulation presented in this paper opens up a new horizon for multiagent AB framework which deserves further investigation.

Acknowledgements

We would like to thank Bob Kowalski, Francesca Toni, Phan Minh Dung for their inspiration and fruitful discussion. Furthermore, the first author would also like to thank Andrei Bondarenko for useful discussion.

References

[Bo92] P. Bonatti (1992) *Model-theoretic semantics for demo* Meta 92.

[BTK93] A. Bondarenko, F.Toni and R.A.Kowalski (1993) *An assumption-based framework for nonmonotonic reasoning* 2nd Logic programming and nonmonotonic reasoning workshop, pp.171-189.

[Du93] P.M. Dung (1993) *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming* IJCAI-93, pp.852-857.

[Ge88] M. Gelfond (1988) *AE logic and formalization of commonsense reasoning* 2nd Int. Workshop on nonmonotonic reasoning.

[Ji93a] Y.J. Jiang (1993a) *Revisable autoepistemic logic, abduction and truth maintenance systems* to appear.

[Ji93b] Y.J. Jiang (1993b) *On the autoepistemic reconstruction of logic programming* New Generation Computing 11.

[Ji94] Y.J. Jiang (1994) *On multiagent autoepistemic logic - an extrospective view* KR 94.

[Ko88] K. Konolige (1988) *On the relationship between default and autoepistemic logic* AI 35.

[Le90] H.J. Levesque (1990) *All I know : A study in autoepistemic logic* AI 42(3), pp. 263-309.

[LS89] F. Lin and Y. Shoham (1989) *Argument Systems:A uniform basis for nonmonotonic reasoning* KR'89 pp.245-255.

[MT91] W. Marek & M. Trunszczynski (1991) *Relating autoepistemic logic and default logic* JACM 38, 1991.

[MD80] D. McDermott & J. Doyle (1980) *Non-monotonic logic I* AI 13, pp. 41-72.

[McD82] D. McDermott (1982) *Nonmonotonic logic II: Nonmonotonic modal theories* JACM 29.

[Mo85] R. C. Moore (1985) *Semantic considerations of nonmonotonic logic* AI 25.

[LJJ91] L.M. Pereira, J.N. Aparico & J.J. Alferes (1991) *Nonmonotonic reasoning with well-founded semantics* ICLP 91.

[Pr91] T.Przymusinski (1991) *3-valued nonmonotonic logics and semantics of logic programs* AI 49.

[Ri78] Reiter (1978) *Closed World Assumption* Logic and Database, edited by Clark and Tarlund.

[Sc91] G. Schwarz (1991) *Autoepistemic logic of knowledge* 1st Logic programming and nonmonotonic reasoning workshop.

Computing queries from prioritized default theories

Torsten Schaub

IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France

Abstract. We develop a method for query-answering from prioritized default theories. This approach allows for computing queries in a prioritized variant of default logic that has been recently proposed by Brewka. This is accomplished by extending an approach to query-answering in conventional default logics based on the idea of structure-oriented theorem proving.

1 Introduction

Reiter's default logic is one of the best known and most-widely used formalisms for default reasoning. In default logic, knowledge is represented by means of a set of closed formulas W and a set of *default rules* of the form $\frac{\alpha : \beta}{\omega}$, where α, β, ω are closed formulas called the prerequisite, the justification and the consequent. The defaults in D induce one (or multiple) *extensions* of the facts in W. Any such extension, E say, is a deductively closed set of formulas containing W such that for any $\frac{\alpha : \beta}{\omega} \in D$, if $\alpha \in E$ and $\neg\beta \notin E$ then $\omega \in E$.

Despite default logic's popularity, we encounter severe deficiencies in the most general setting. For instance, default logic does not guarantee the existence of extensions. Moreover, it does not address the notion of *specificity*, a fundamental principle in commonsense reasoning according to which more specific defaults should be preferred over less specific ones. For avoiding many problems, it is usually sufficient to restrict oneself to certain subclasses of default logic like for instance *normal default theories*, consisting of defaults of the form $\frac{\alpha : \beta}{\beta}$. Unfortunately, such a restriction does not work for addressing the failure of specificity. Consider the statements "students are typically adults", "adults are typically employed", and "students are typically not employed" along with a student, which yields the following default theory:

$$\left(\left\{\frac{S : \neg E}{\neg E}, \frac{S : A}{A}, \frac{A : E}{E}\right\}, \{S\}\right) \qquad (1)$$

Intuitively, we would like to derive A and $\neg E$. We do not want to derive E, since this violates the principle of specificity. In fact, being a student is more specific than being an adult and so we want to give the first default priority over the third one. Yet theory (1) gives two extensions, $Th(\{A, \neg E, S\})$ and $Th(\{A, E, S\})$, showing that even normal default theories fail to address specificity.

Recently, a couple of approaches were developed for isolating specificity information (cf. [2, 3]) out of a given set of defaults. In the above example these methods allow us to detect that the default $\frac{S : \neg E}{\neg E}$ is more specific than the default $\frac{A : E}{E}$. In this way, [2] obtains so-called *prioritized normal default theories*, ie. normal default theories along with a strict partial order on the defaults. This approach eliminates the second extension of default theory (1) and we obtain only the first, more specific extension. So far, there is however no way of computing queries from prioritized normal default theories. We address this gap in what follows by extending a method for query-answering in default logics recently proposed in [4] and further elaborated in [5]. The basic idea behind this approach is to treat defaults as classical implications along with some qualifying conditions restricting the use of such rules in the course of the proof search. To this end, query-answering in default logic is decomposed into classical deduction plus additional restrictions on proofs ensuring the characteristics of defaults. The restriction of classical proofs is supported by the notion of structure-sensitive theorem proving as found in the connection method [1].

In what follows, we modify and extend the approach of [4] in order to allow for query-answering from prioritized default theories. This is accomplished by identifying the characteristic features of Brewka's approach to integrating specificity into default logic. Then, we map these features onto restrictions on classical proofs in the connection method. In this way, we obtain a connection calculus for query-answering from prioritized default theories.

2 Prioritized default logic

As defined in [2], a prioritized default theory is a triple $(D, W, <)$, where D is a finite set of normal defaults,[1] W a set of formulas, and $<$ a strict partial order on D. As mentioned in the introduction, a more specific default δ gets priority over a less specific default δ'; this is expressed by $\delta < \delta'$. The idea is as follows [2]: *"during the generation of an extension defaults with higher priorities whose prerequisite has been derived are always considered before defaults of lower priority. This guarantees that defaults of lower priority are applied only if this does not lead to a conflict with a 'better' default. Since the partial ordering usually leaves some priorities undecided we have to take every total ordering of the defaults into account that respects the original partial order. Every such total order will give rise to one extension. Different orderings may generate the same extension."*

Now, a central concept is that of *activeness*: A normal default rule δ is active in a set of formulas E iff $\mathit{Prereq}(\delta) \in E$, $\neg \mathit{Conseq}(\delta) \notin E$, and $\mathit{Conseq}(\delta) \notin E$.[2] With this concept, any total extension $\ll$ of $<$ defines a prioritized extension of a prioritized default theory $(D, W, <)$ in the following way.

Definition 2.1 [2] *Let $(D, W, <)$ be a prioritized default theory and let $\ll$ be a strict total order containing $<$. Define $E_0 = Th(W)$ and for $i \geq 0$*

$$E_{i+1} = \begin{cases} E_i & \textit{if there is no } \delta \in D \textit{ is active in } E_i \text{ ,} \\ Th(E_i \cup \{\mathit{Conseq}(\delta)\}) & \textit{otherwise, where } \delta \textit{ is the } \ll\textit{-minimal} \\ & \textit{default rule that is active in } E_i. \end{cases}$$

Then, E is the prioritized extension of $(D, W, <)$ iff $E = \bigcup_{i \geq 0} E_i$

Consider our initial example in (1). Using the formalisms in [2, 3], we obtain a precedence between the first and the third default: $\frac{S : \neg E}{\neg E} < \frac{A : E}{E}$. This yields

[1] As argued in [2], priorities compensate the additional expressiveness of non-normal defaults so that he confines himself to normal ones.

[2] For convenience, we denote the prerequisite of a default δ by $\mathit{Prereq}(\delta)$ and its consequent by $\mathit{Conseq}(\delta)$.

the following prioritized default theory:

$$\left(\left\{\frac{S:\neg E}{\neg E},\frac{S:A}{A},\frac{A:E}{E}\right\},\{S\},\left\{\frac{S:\neg E}{\neg E}<\frac{A:E}{E}\right\}\right) \tag{2}$$

From the order given above, we obtain three total orders respecting <, namely

$$\delta_2 \ll \delta_1 \ll \delta_3, \qquad \delta_1 \ll \delta_2 \ll \delta_3, \qquad \delta_1 \ll \delta_3 \ll \delta_2.$$

For brevity, we denote the defaults in (2) by $\delta_1, \delta_2, \delta_3$ from left to right. It is easy to verify that for each total order, we obtain the first and more specific extension, $Th(\{A, \neg E, S\})$, as the only prioritized extension. Consider the first of the three total orders. This yields the following sequence of partial prioritized extensions: $E_0 = Th(\{S\})$, $E_1 = Th(\{S, A\})$, $E_2 = Th(\{S, A, \neg E\})$, $E_3 = E_2$, ... The two other orders lead to the same extension. Thus in all three cases, the default $\frac{S:\neg E}{\neg E}$ takes priority over the less specific default $\frac{A:E}{E}$.

In what follows, we want to identify the characteristic features of Brewka's approach. These characteristics will then be mapped onto their computational counterparts in Section 4. The two general features distinguishing defaults from classical implications are the *character of an inference rule* and the additional *consistency check*. While the latter is handled in default logic in the usual manner by testing satisfiability, the former needs a more subtle treatment: The character of an inference rule can be captured by the notion of *groundedness*. We call a set of defaults D *grounded* in a set of facts W iff there exists an enumeration $\langle\delta_i\rangle_{i\in I}$ of D such that for $i \in I$

$$W \cup Conseq(\{\delta_0, \dots, \delta_{i-1}\}) \vdash Prereq(\delta_i) \tag{3}$$

What is the additional qualifying condition needed for incorporating priorities? We have seen above that the nature of an inference rule is reflected by the property of groundedness, which relies on forming sequences of defaults. Now, the idea is to isolate sequences of defaults that respect the given partial order. In contrast to the independence of the concepts of groundedness and consistency, however, we encounter a more difficult situation if we want to preserve certain priorities. This is so because a priority between defaults is only effective if both defaults are applicable, ie. if their prerequisites are derivable and their consequents are consistent. In this way, the preservation of priorities becomes dependent on the concepts of groundedness and consistency. This leads us to following concept of *activeness* for sequences of defaults.

Definition 2.2 *Let $(D, W, <)$ be a prioritized default theory and let $i \in I$ for some index set I of D. Then, a default rule δ is active at i in I iff*

1. $W \cup Conseq(\{\delta_0, \dots, \delta_{i-1}\}) \vdash Prereq(\delta)$,
2. $W \cup Conseq(\{\delta_0, \dots, \delta_{i-1}\}) \nvdash \neg Conseq(\delta)$.

This definition differs from the one in [2] mainly in the way previously applied defaults are ignored. In [2], this is done by the condition $Conseq(\delta) \notin E$, whereas we use the structure provided by a sequence of defaults. That is, we require that an active default does not belong to the sequence at hand (see Theorem 2.1 below). The advantage of the latter definition is that it explicates the underlying concepts of groundedness and consistency. In fact, Condition *(1)* is identical to the condition given for groundedness in (3). In contrast to the global notion of consistency employed for regular default logic, however, consistency is incrementally verified in Condition *(2)*. With this conception we obtain the following alternative characterization of prioritized extensions on which our approach to query-answering relies.

Theorem 2.1 *Let $(D, W, <)$ be a prioritized default theory and let E be a set of formulas. Then, E is a prioritized extension of $(D, W, <)$ iff $E = Th(W \cup Conseq(D'))$ for a maximal $D' \subseteq D$ such that there exists an enumeration $\langle \delta_i \rangle_{i \in I}$ of D' where we have for $i \in I$ that δ_i is a $<$-minimal default among all defaults in $D \setminus \{\delta_0, \ldots, \delta_{i-1}\}$ that are active at i in I.*

The proof of this and all subsequent theorems are given in [5]. This characterization strongly relies on the notion of a sequence of defaults. Apart from the aforementioned differences among the notions of activeness, this definition has the advantage that it does not rely on total extensions of the underlying order. Thus, the need for computing a total extension $\ll$ of an order $<$ for computing prioritized extension is dropped.

Consider our student example along with the partial order $<$ illustrated in Figure 1. Starting from the facts $\{S\}$, we get two active defaults, δ_1 and δ_2. This is indicated by means of filled circles instead of plain circles next to the defaults. Since neither is preferred over the other one, both constitute $<$-minimal defaults and we can apply either of them. Choosing δ_2 renders it inactive and activates δ_3, since now its prerequisite is derivable from $\{S\} \cup \{A\}$ (ie. the facts along with the consequent of δ_2). Since δ_1 is still active, we obtain two active defaults, δ_1 and δ_3. Now, δ_1 has to be applied since it is preferred over δ_3 by $\delta_1 < \delta_3$. Finally, no default is active any more and we obtain the enumeration $\langle \delta_2, \delta_1 \rangle$, inducing a prioritized extension containing $\{S\} \cup \{A, \neg E\}$.

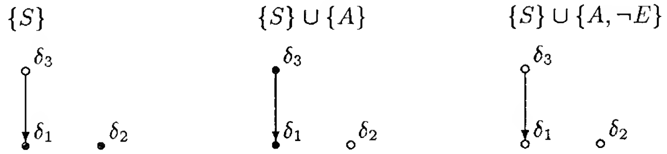


Fig. 1. Computing the prioritized extension of default theory (2).

This suggests the following approach to query-answering. In order to verify whether a formula φ is in some prioritized extension E of a prioritized default theory $(D, W, <)$, we have to find a subset of D that allows for deriving φ and complies with the above requirements. This is made precise in the following corollary to Theorem 2.1.

Corollary 2.2 *Let $(D, W, <)$ be a prioritized default theory and let E be a set of formulas. Then, $\varphi \in E$ for some prioritized extension E of $(D, W, <)$ iff $W \cup Conseq(D') \vdash \varphi$ for some $D' \subseteq D$ such that there exists an enumeration $\langle \delta_i \rangle_{i \in I}$ of D' where we have for $i \in I$ that δ_i is a $<$-minimal default among all defaults in $D \setminus \{\delta_0, \ldots, \delta_{i-1}\}$ that are active at i in I.*

Note that the activeness of defaults has to be checked wrt all non-applied defaults, viz. $D \setminus \{\delta_0, \ldots, \delta_{i-1}\}$. Query-answering is accomplished as sketched above, with the important exception that the sequence of defaults need not be maximal. That is, for proving A, we can stop after the second step in Figure 1.

In the sequel, we follow [4] in dealing with default theories in atomic format: For a default theory (D, W) over $\mathcal{L}_\Sigma$, let $\mathcal{L}_{\Sigma'}$ be the language obtained by adding the new propositions $\alpha_\delta, \beta_\delta$ for each $\delta \in D$. Then, a default theory (D, W) over $\mathcal{L}_\Sigma$ is mapped onto a default theory (D', W') over $\mathcal{L}_{\Sigma'}$, where

$$D' = \left\{ \tfrac{\alpha_\delta : \beta_\delta}{\beta_\delta} \,\middle|\, \delta \in D \right\} \quad W' = W \cup \{ \mathit{Prereq}(\delta) \rightarrow \alpha_\delta, \beta_\delta \rightarrow \mathit{Conseq}(\delta) \mid \delta \in D \}.$$

The resulting default theory (D', W') is called the *atomic format* of the original default theory (D, W). As shown in [4], this transformation is a conservative extension of the formalism.

3 Default theorem proving in the connection method

We start by describing the approach for query-answering in default logics developed in [4]. This approach is based on the connection method [1], which allows for testing the unsatisfiability of formulas in conjunctive normal form (CNF). Unlike resolution-based methods that decompose formulas in order to derive a contradiction, the connection method analyses the structure of formulas for proving their unsatisfiability. This structure-sensitive nature allows for an elegant characterization of proofs in default logic, as we see below.

In the connection method, formulas in CNF are displayed two-dimensionally in the form of *matrices* (see (4)). A matrix is a set of sets of literals (literal occurrences, to be precise).[3] Each column of a matrix represents a *clause* of the CNF of a formula. In order to show that a sentence φ is entailed by a sentence W, we prove that $W \wedge \neg\varphi$ is unsatisfiable. In the connection method this is accomplished by path checking: A *path* through a matrix is a set of literals, one from each clause. A *connection* is an unordered pair of literals which are identical except for the negation sign (and possible indices). A *mating* is a set of connections. A mating *spans* a matrix if each path through the matrix contains a connection from the mating. Finally, a formula, like $W \wedge \neg\varphi$, is unsatisfiable iff there is a spanning mating for its matrix.

The approach relies on the idea that a default can be decomposed into a classical implication along with two qualifying conditions, one accounting for the character of an inference rule and another one enforcing the respective consistency conditions. The computational counterparts of these qualifying conditions are given by the proof-oriented concepts of *admissibility* and *compatibility*, which we introduce in the sequel. We deal with a propositional language over a finite alphabet.

In order to find out whether a formula φ is in some extension of a default theory (D, W) we proceed as in [4]. First, we transform the defaults in D into their sentential counterparts. This yields a set of indexed implications

$$W_D = \left\{ \alpha_\delta \rightarrow \beta_\delta \,\middle|\, \tfrac{\alpha_\delta : \beta_\delta}{\beta_\delta} \in D \right\}.$$

[3] In the sequel, we simply say literal instead of literal occurrences; the latter allow for distinguishing between identical literals in different clauses.

Second, we transform both W and W_D into their clausal forms, C_W and C_D. The clauses in C_D, like $\{\neg\alpha_\delta, \beta_\delta\}$, are called *$\delta$-clauses;* all other clauses like those in C_W are referred to as *ω-clauses.* Now, we are ready for query-answering. That is, φ is derivable from (D, W) iff there is a spanning mating for the matrix $C_W \cup C_D \cup \{\neg\varphi\}$ agreeing with the concepts of admissibility and compatibility.[4]

Consider our student example. The encoding of the set of defaults yields the following set, W_D, of implications: $\{S_{\delta_1} \rightarrow \neg E_{\delta_1}, S_{\delta_2} \rightarrow A_{\delta_2}, A_{\delta_3} \rightarrow E_{\delta_3}.\}$ For illustration, let us verify that we can in fact derive that a student is employed, E, from our initial default theory (1). Recall that this clashes with our intuitions on preferring the more specific extension and that E is not derivable from prioritized default theory (2). We first transform the fact S and the implications in W_D into their clausal form. The resulting clauses are given two-dimensionally as the first four columns of the matrix in (4). The full matrix is obtained by adding the clause containing the negated query, $\neg E$. In fact, the matrix has a spanning mating, $\{\{S, \neg S_{\delta_2}\}, \{A_{\delta_2}, \neg A_{\delta_3}\}, \{E_{\delta_3}, \neg E\}\}$, whose connections are indicated as arcs linking the respective literals.

$$\left[\begin{array}{ccccc} & \neg S_{\delta_1} & \neg S_{\delta_2} & \neg A_{\delta_3} & \neg E \\ S & \neg E_{\delta_1} & A_{\delta_2} & E_{\delta_3} & \end{array}\right] \tag{4}$$

The above matrix illustrates that not all of the clauses are necessarily involved in providing a spanning mating for a matrix. A useful concept is then that of a *core* of a matrix M wrt a mating Π, which allows for isolating the clauses relevant to the underlying proof. We define the core of M wrt Π as in [4] as[5]

$$\kappa(M, \Pi) = \{c \in M \mid \exists \pi \in \Pi \; . \; c \cap \pi \neq \emptyset\} \, .$$

For instance, the core of the preceding matrix relative to the drawn mating is given by the first and the last three clauses.

In default logic, the nature of an inference rule is reflected by the property of groundedness, which relies on forming sequences of defaults. In fact, the connection method allows for imposing a similar restriction on the clausal counterparts of defaults. This leads to the first qualifying condition on proofs given by the concept of admissibility [4].

Definition 3.1 *Let Π be a mating for $C_W \cup C_D$ where C_W and C_D are sets of ω- and δ-clauses. Then, $(C_W \cup C_D, \Pi)$ is admissible iff there is an enumeration $\langle\{\neg\alpha_{\delta_i}, \beta_{\delta_i}\}\rangle_{i \in I}$ of $\kappa(C_D, \Pi)$ such that for $i \in I$, Π is a spanning mating for*

$$C_W \cup \left(\bigcup_{j=0}^{i-1}\{\{\neg\alpha_{\delta_j}, \beta_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_i}\}\}. \tag{5}$$

We say that $(C_W \cup C_D, \Pi)$ is admissible at i in an index set I, if (5) holds for $i \in I$. Consider proof (4) in our student example. There, we obtain the enumeration $\langle\{\neg S_{\delta_2}, A_{\delta_2}\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\rangle$, which in turn leads to the following matrices; each representing a set of clauses as specified in (5):

$$\left[\begin{array}{cc} & \neg S_{\delta_2} \\ S & \end{array}\right] \qquad \left[\begin{array}{ccc} & \neg S_{\delta_2} & \neg A_{\delta_3} \\ S & A_{\delta_2} & \end{array}\right] \tag{6}$$

[4] Without loss of generality, we deal with atomic queries only, since any query can be transformed into "atomic format" too.

[5] Recall that we deal with literal occurrences.

Observe that the preceding matrices are in fact submatrices of matrix (4). Apparently, each of these submatrices has a spanning mating, so that the original matrix along with its mating, given in (4), constitutes an admissible proof.

The second qualifying condition for proofs is given in [4] by the concept of compatibility.[6] Unfortunately, the "global" notion of compatibility used in [4] is inappropriate[6] for query-answering in prioritized default logic. Rather the specification of activeness given in Definition 2.2 suggests an incremental approach in which compatibility is gradually verified each time a default applies. This motivates the following definition introducing the concept of incremental compatibility.

Definition 3.2 (Incremental Compatibility) *Let Π be a mating for $C_W \cup C_D$ where C_W and C_D are sets of ω- and δ-clauses. Let $\langle\{\neg\alpha_{\delta_i}, \beta_{\delta_i}\}\rangle_{i\in I}$ be an enumeration of $\kappa(C_D, \Pi)$. Then, $(C_W \cup C_D, \Pi)$ is incrementally compatible wrt I iff for $i \in I$, there is no spanning mating for*

$$C_W \cup \left(\bigcup_{j=0}^{i-1}\{\{\beta_{\delta_j}\}\}\right) \cup \{\{\beta_{\delta_i}\}\}. \tag{7}$$

We say that $(C_W \cup C_D, \Pi)$ is compatible at i in an index set I, if (7) holds for $i \in I$.

For "admissible matrices", incremental compatibility can be verified using δ-clauses like $\{\neg\alpha_\delta, \beta_\delta\}$ instead of clauses containing merely the consequent of a default, $\{\beta_\delta\}$.

Theorem 3.1 *Let Π be a mating for $C_W \cup C_D$, where C_W and C_D are sets of ω- and δ-clauses, such that $(C_W \cup C_D, \Pi)$ is admissible. Then, Π is a spanning mating for $C_W \cup C_D$ iff Π is a spanning mating for $C_W \cup C_J$ where $C_J = \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \beta_\delta\} \in C_D\}$.*

This offers the computational advantage that we can verify compatibility on (almost) the same matrices as used for verifying admissibility. The latter is checked on matrices of the form $C_W \cup \left(\bigcup_{j=0}^{i-1}\{\{\neg\alpha_{\delta_j}, \beta_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_i}\}\}$. For admissibility, all path through such a matrix have to be complementary. For compatibility, there has to emerge a non-complementary path if we replace the clause $\{\{\neg\alpha_{\delta_i}\}\}$ by the clause $\{\{\neg\alpha_{\delta_i}, \beta_{\delta_i}\}\}$; simply by adding the consequent of the default δ_i.

Consider the submatrices used in (6) for verifying admissibility. In fact, we can (re)use these matrices for testing compatibility. This amounts to verifying compatibility for the following supermatrices of those in (6).

$$\begin{bmatrix} & \neg S_{\delta_2} \\ S & A_{\delta_2} \end{bmatrix} \qquad \begin{bmatrix} & \neg S_{\delta_2} & \neg A_{\delta_3} \\ S & A_{\delta_2} & E_{\delta_3} \end{bmatrix} \tag{8}$$

We start by verifying whether all paths through the matrix $\{\{S\}, \{\neg S_{\delta_2}\}\}$ are complementary. Since this is the case the left matrix is admissible. For checking its compatibility, we merely have to look for a non-complementary path through the facts, here $\{\{S\}\}$ and the consequent of δ_2, A_{δ_2}. All other path are complementary by admissibility. In fact, the path $\{S, A_{\delta_2}\}$ is not complementary and so

[6] Due to severe space limitations, we have to refer the reader to [5] for details.

the left matrix is compatible. Now, for verifying admissibility in the case of the right matrix (cf. (6)), we make use of the information gathered on the left matrix, which is in fact a submatrix of the right one. In this example, it is enough to check whether adding the prerequisite of δ_3, $\neg A_{\delta_3}$ makes all non-complementary path in the left matrix complementary. In fact, this is the case since $\{S, A_{\delta_2}\}$ is the only non-complementary path in the left matrix and $\{S, A_{\delta_2}\} \cup \{\neg A_{\delta_3}\}$ is complementary. The compatibility of the right matrix is established in analogy to that of the left matrix; again by reusing the information gathered while verifying its admissibility.

4 Priority preserving default proofs

We now turn to query-answering from prioritized default theories. For this, we turn the notion of activeness defined in Definition 2.2 into its "computational" counterpart. The order on defaults extends to δ-clauses in the obvious way. That is, $\delta < \delta'$ iff $\{\neg\alpha_\delta, \beta_\delta\} < \{\neg\alpha_{\delta'}, \gamma_{\delta'}\}$.

Definition 4.1 (Activeness) *Let Π be a mating for $C_W \cup C_D$ where C_W and C_D are sets of ω- and δ-clauses. Let $i \in I$ for some index set I of C_D. Then, a δ-clause $\{\neg\alpha_\delta, \beta_\delta\}$ is active at i in I iff*

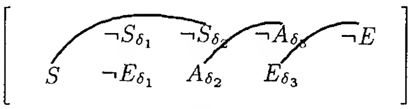
1. *$(C_W \cup C_D, \Pi)$ is admissible at i in I,*
2. *$(C_W \cup C_D, \Pi)$ is compatible at i in I.*

Notice that the concept of activeness is in fact parameterized by C_W and C_D.

With this notion of activeness, we can now define the concept of "priority preserving default proofs":

Definition 4.2 (Priority preserving) *Let Π be a mating for $C_W \cup C_D$ where C_W and C_D are sets of ω- and δ-clauses and let $<$ be a strict partial order on C_D. Then, $(C_W \cup C_D, \Pi)$ is $<$-priority preserving iff there is an enumeration $\langle c_i\rangle_{i\in I}$ of $\kappa(C_D, \Pi)$ such that for $i \in I$, we have that c_i is a $<$-minimal δ-clause among all δ-clauses in $C_D \setminus \bigcup_{j<i}\{c_j\}$ that are active at i in I.*

This definition may be nicely illustrated by of the proof given in (4). There, we have proved E (ie. student are employed) from our initial default theory (1). This has resulted in the following "default proof":

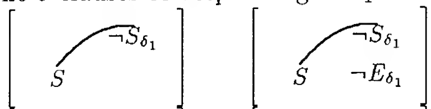


In what follows, we demonstrate that this proof is *not* priority preserving for the order given in Section 2, namely $\frac{S:\neg E}{\neg E} < \frac{A:E}{E}$. This corresponds to the result obtained in prioritized default logic, where E is not derivable.

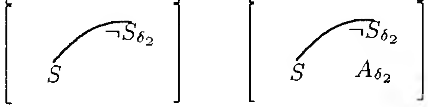
The above default proof yields the enumeration $\langle\{\neg S_{\delta_2}, A_{\delta_2}\}\{\neg A_{\delta_3}, E_{\delta_3}\}\rangle$ of δ-clauses. In order to verify whether the default proof is priority preserving, we have to check whether the clause $\{\neg S_{\delta_2}, A_{\delta_2}\}$ is a $<$-minimal δ-clause active in the given facts.[7] Actually, there are two $<$-minimal δ-clause that are active at

[7] For simplicity, we talk explicitly about the referenced set of formulas or clauses, respectively, instead of giving the corresponding indexes.

this stage; these are the δ-clauses corresponding to δ_1 and δ_2:



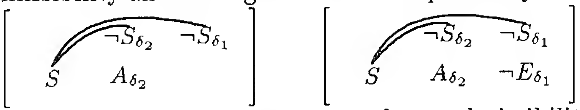
(9)



(10)

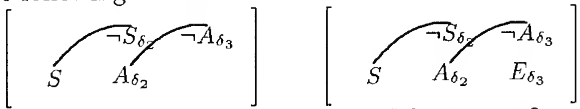
The matrices on the left show that both δ-clauses confirm admissibility (since both are complementary) and the matrices on the right show the same for compatibility (since both contain a non-complementary path). Therefore both δ-clauses are active in the set of ω-clauses $\{\{S\}\}$. Since the δ-clauses corresponding to δ_1 and δ_2 are active and neither of them is preferred over the other one, it follows that both constitute $<$-minimal δ-clauses. Hence, $\{\neg S_{\delta_2}, A_{\delta_2}\}$ is a $<$-minimal default rule active in $\{\{S\}\}$.

Now, we have to verify whether the δ-clause $\{\neg A_{\delta_3}, E_{\delta_3}\}$ is $<$-minimal among the δ-clauses active in $\{\{S\}, \{\neg S_{\delta_2}, A_{\delta_2}\}\}$. First of all, we observe that the δ-clause corresponding to δ_1 is still active in the matrix extended by the δ-clause $\{\neg S_{\delta_2}, A_{\delta_2}\}$. This is shown in the following two matrices. As above the left matrix accounts for admissibility and the right one for compatibility.



(11)

The left matrix is complementary and thus confirms admissibility, whereas the right one possesses a non-complementary path, $\{S, A_{\delta_2}, \neg E_{\delta_1}\}$, confirming compatibility. In fact, the δ-clause $\{\neg A_{\delta_3}, E_{\delta_3}\}$ is also active in $\{\{S\}, \{\neg S_{\delta_2}, A_{\delta_2}\}\}$, as shown by the following two matrices.



(12)

As above, the left matrix is complementary and hence confirms admissibility, whereas the right one has a non-complementary path, $\{S, A_{\delta_2}, E_{\delta_3}\}$, confirming compatibility. Hence, both δ-clauses, $\{\neg S_{\delta_2}, A_{\delta_2}\}$ and $\{\neg A_{\delta_3}, E_{\delta_3}\}$, are active in $\{\{S\}, \{\neg S_{\delta_2}, A_{\delta_2}\}\}$. However, $\{\neg A_{\delta_3}, E_{\delta_3}\}$ is not $<$-minimal among the active δ-clauses. This is so because $\delta_1 < \delta_3$. Hence, we have shown that default proof (4) for E is not priority preserving. This corresponds to the result obtained in prioritized default logic. In fact, we could have saved the test whether δ_1 is active after establishing activeness for δ_3, since $\delta_1 < \delta_3$. The discard of defaults that are $<$-greater than active defaults is a general way of cutting down the number of choices to be considered at each stage.

In general, we have the following result showing that our method is sound and complete for prioritized default logic:

Theorem 4.1 *Let $(D, W, <)$ be a prioritized default theory in atomic format and let $W_D = \left\{\alpha_\delta \rightarrow \beta_\delta \;\middle|\; \frac{\alpha_\delta : \beta_\delta}{\beta_\delta} \in D\right\}$ and φ an atomic formula. Then, $\varphi \in E$ for some prioritized extension E of $(D, W, <)$ iff there is a spanning mating Π for the matrix M of $W \cup W_D \cup \{\neg\varphi\}$ such that (M, Π) is $<$-priority preserving.*

Finally, we show that we can indeed prove that students are not employed from prioritized default theory (2). Recall that we obtained the prioritized extension $Th(\{A, \neg E, S\})$. For proving $\neg E$, we have to replace the last clause in (4) by the clause $\{E\}$ containing the negated query. This yields the following default proof:

$$\left[\begin{array}{ccccc} & \neg S_{\delta_1} & \neg S_{\delta_2} & \neg A_{\delta_3} & E \\ S & \neg E_{\delta_1} & A_{\delta_2} & E_{\delta_3} & \end{array}\right] \tag{13}$$

Here, we obtain the enumeration $\langle\{\neg S_{\delta_1}, \neg E_{\delta_1}\}\rangle$. As shown in (9), $\{\neg S_{\delta_1}, \neg E_{\delta_1}\}$ is a $<$-minimal δ-clause active in $\{\{S\}\}$. This shows that the proof in (13) is priority preserving and hence is $\neg E$ in the prioritized extension of the prioritized default theory in (2).

An algorithm for computing queries from prioritized default theories is given in [5]. Due to severe space limitations this has been omitted here.

5 Conclusion

We proposed a new method along with a declarative algorithm for query-answering from prioritized default theories. This was accomplished by extending and modifying an existing approach to query-answering in conventional default logics based on the idea of structure-oriented theorem proving. First, we refined the approach of [2] to prioritized default logic by giving an alternative characterization of prioritized extensions. This characterization is based on sequences of defaults and avoids an explicit check for already applied defaults and the computation of total orders. Second, we extended the approach of [4] by the concept of incremental compatibility. We showed how this property can be verified in connection with admissibility. Finally, we gradually mapped the concepts developed for prioritized default logic onto a new method for query-answering from prioritized default theories. This resulted in a connection calculus based on the notion of priority preserving matings.

Acknowledgments I would like to thank G. Brewka and A. Rothschild for many discussions. This research was supported by the CEC under grant no. ERB4001GT922433.

References

1. W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, 1987.
2. G. Brewka. Adding priorities and specificity to default logic. In L. Pereira and D. Pearce, editors, *Proc. of JELIA'94.* Springer, 1994. To appear.
3. J. Delgrande and T. Schaub. A general approach to specificity in default reasoning. In J. Doyle, P. Torasso, and E. Sandewall, eds, *Proc. of KR'94*, p. 146–157. Morgan Kaufmann, 1994.
4. A. Rothschild and T. Schaub. A computational approach to default logics. In G. Brewka and C. Witteveen, eds, *Dutch/German Workshop on Nonmonotonic Reasoning*, p. 1–14, 1993.
5. T. Schaub. A new methodology for query-answering in default logics via structure-oriented theorem proving. Technical report, IRISA, 1994. Forthcoming.

Beliefs and Bilattices

Kwang Mong SIM

Knowledge Science Institute, University of Calgary
Calgary, Alberta, Canada T2N 1N4
kwang@cpsc.ucalgary.ca

Abstract. A *bilattice* is a structure which can be viewed as a class of truth values that can accommodate incomplete and inconsistent information. In bilattice theory, knowledge are ordered along two dimensions : truth/ falsity and certainty/ uncertainty. Bilattice theory has been applied in areas such as *non-monotonic reasoning*, *truth maintenance systems* and more recently to logic programming. In this paper, I will attempt to connect bilattice theory to epistemic logic. In the last decade there has been a resurgence of work on epistemic logic in artificial intelligence. Among them were the logic of *implicit and explicit beliefs* and the logic of *awareness* . Most of these approaches attempted to alleviate the problem of *logical omniscience*. Although these different models have surface dissimilarities, a closer examination shows that they have strong resemblance. Further, given the current proliferation of epistemic logic, it seems prudent to establish comparisons among the various approaches and to consider the issue of unification among them. The main contribution of this research is to show that a uniform framework for reasoning about knowledge can be defined in the context of bilattice theory. This formulation is based on the observation that epistemic notions such as implicit belief, explicit belief and awareness can be associated with subsets of truth values in a bilattice. It is hoped that the proposed work will lead to new insights in both bilattice theory and epistemic logic.

1 Introduction

It has been recognized [16, 2] that classical two-valued logic is inadequate for many applications and restricting to a true-false dichotomy is unnecessary. In fact, in some artificial intelligence applications, it is even necessary to employ a many-valued logic which can admit incomplete information [12] and even conflicting information [2]. In the literature on multi-valued logic, several researchers such as Belnap [2] and Ginsberg [8] have employed an algebraic approach to many-valued logic based on DeMorgan lattices and have shown the applications of these structures. Belnap advocates the use of multi-valued logic as a general means of handling inconsistency by adding a fourth value (*overdetermined*) to Lukasiewicz's [15] three-valued logic. He has also observed that a lattice structure can be put upon these four truth values. In fact, this lattice structure is the simplest example of a *bilattice*. Ginsberg's bilattice theory emphasizes the need to attach varying degrees of truth and certainty to logical sentences. Bilattice theory has been applied in areas such as *non-monotonic reasoning* [7] and *truth maintenance systems* [4]. More recently, Fitting [6] has shown the semantics of distributed logic programs can be defined relative to a class of bilattices. This research attempts to make connections between bilattice theory and several logics for reasoning about knowledge.

Reasoning about knowledge and beliefs was first studied in philosophy under the topics of *epistemic* (knowledge) and *doxastic* (belief) logic [10]. Hintikka [10] was the first to propose the idea of applying *possible-worlds* semantics to model knowledge and beliefs. The intuitive idea is that beside the true state of affairs (actual worlds), there are other possible states of affairs. Under this interpretation, an agent is said to *know* (or *belief*) a fact α if α is true in all the states that it considers possible. Although the possible-worlds approach can be customized to capture different properties of knowledge by making slight alterations to its semantics [9], it suffers from what Hintikka [11] termed the *logical omniscience* problem. Logical omniscience requires an agent to know all logical consequences of its beliefs and all valid sentences. For example, if an agent knows a set of formulas Γ and if Γ logically implies the formula α then the agent will also know α. Additionally, an agent cannot hold inconsistent beliefs without believing every sentence. These problems result in intractability because the agent is required to compute all logical consequences of its beliefs. In practice, resource-bounded agents usually do not have enough time or memory to derive an *explicit* representation of each fact.

Objectives : In the last decade, there has been a resurgence of work to mitigate the logical omniscience problem. Even though these logics have surface dissimilarities, a closer examination shows that they have strong resemblance. Further, given the current proliferation of epistemic logic, it seems prudent to establish comparisons among the various approaches and to consider the issue of unification among them. More explicitly, the objective of this research is to formulate a uniform framework that harmonizes several existing approaches (in particular the logic of *implicit and explicit belief* and the logic of *awareness*). Although it is noted that there has been previous attempt [18] to formulate a unifying model for epistemic logic, the main contribution of this research is to show that a uniform framework for reasoning about knowledge can be defined in the context of bilattice theory. This formulation is based on the observation that epistemic notions such as implicit belief, explicit belief and awareness can be associated with subsets of truth values in a *bilattice*. A brief exposition of bilattice theory will be given in section 3, while the logic of implicit and explicit belief and the logic of awareness are presented in section 2. The original contribution of this research is reported in section 4.

2 Epistemic Logics in Artificial Intelligence

2.1 Logic of Implicit and Explicit Belief

Levesque's [14] logic of implicit and explicit belief was one of the earliest work in artificial intelligence which addressed the problem of logical omniscience. In his model, Levesque defined *explicit beliefs* as those sentences actively held by an agent and *implicit beliefs* as logical consequences of its explicit beliefs. He employed a *situation semantics* [1] for modeling the beliefs of an agent. This approach avoids some aspects of logical omniscience by admitting situations (worlds) that are incomplete and incoherent. This is achieved by decoupling the semantics of a formula with two independent notions of truth support and falsity support.

Subsequently, a model for implicit and explicit belief is a tuple $\mathbf{M_L} = <\mathbf{S}, \Sigma, \mathbf{T}, \mathbf{F}>$ where **S** is a set of all situations and Σ is any subset of **S**. **T** and **F** are functions from **P** (a set of atomic sentences) to **S**, such that for any $p \in \mathbf{P}$, $\mathbf{T}(p)$ are those situations that support the truth of p and $\mathbf{F}(p)$ are those that support the falsity of p. A situation is incoherent if it is in the set $\mathbf{T}(p) \cap \mathbf{F}(p)$; its incomplete if it is not in $\mathbf{T}(p) \cup \mathbf{F}(p)$. A situation that is not incomplete or incoherent is called a possible world. A possible world s is compatible with another situation $s`$ iff $s \in \mathbf{T}(p)$ then $s` \in \mathbf{T}(p)$ and $s \in \mathbf{F}(p)$ then $s` \in \mathbf{F}(p)$, for each primitive atom p. There are two support relations $\models_T$ and $\models_F$ between situations and formulas from **PL. PL** is a language that is formed using a set of atomic letters from **P** using the standard connectives $\wedge, \neg, \supset, \vee, \equiv$ and two epistemic operators **L** (implicit belief) and **B** (explicit belief); the only restriction is that sentences containing **B** or **L** cannot occur within the scope of the two epistemic operators. Thus, for all $\varphi \in$ **PL**, $\mathbf{M_L}, s \models_T \varphi$ means that s supports the truth of φ and $\mathbf{M_L}, s \models_F \varphi$ mean that s supports the falsity of φ. In [14] the following formal semantics are defined :

(L-p T) $\quad \mathbf{M_L}, s \models_T p$ iff $s \in \mathbf{T}(p)$

(L-p F) $\quad \mathbf{M_L}, s \models_F p$ iff $s \in \mathbf{F}(p)$

(L-$\neg$T) $\quad \mathbf{M_L}, s \models_T \neg\alpha$ iff $\mathbf{M_L}, s \models_F \alpha$

(L-$\neg$F) $\quad \mathbf{M_L}, s \models_F \neg\alpha$ iff $\mathbf{M_L}, s \models_T \alpha$

(L-$\wedge$T) $\quad \mathbf{M_L}, s \models_T \alpha \wedge \beta$ iff $\mathbf{M_L}, s \models_T \alpha$ and $\mathbf{M_L}, s \models_T \beta$

(L-$\wedge$F) $\quad \mathbf{M_L}, s \models_F \alpha \wedge \beta$ iff $\mathbf{M_L}, s \models_F \alpha$ or $\mathbf{M_L}, s \models_F \beta$

(L-BT) $\quad \mathbf{M_L}, s \models_T \mathbf{B}\alpha$ iff $\mathbf{M_L}, t \models_T \alpha$ for all $t \in \Sigma$

(L-BF) $\quad \mathbf{M_L}, s \models_F \mathbf{B}\alpha$ iff $\mathbf{M_L}, s \not\models_T \mathbf{B}\alpha$.

(L-LT) $\quad \mathbf{M_L}, s \models_T \mathbf{L}\alpha$ iff $\mathbf{M_L}, t \models_T \alpha$ for all $t \in W(\Sigma)$

(L-LF) $\quad \mathbf{M_L}, s \models_F \mathbf{L}\alpha$ iff $\mathbf{M_L}, s \not\models_T \mathbf{L}\alpha$.

Implicit Belief : In (L-LT), $W(\Sigma)$ is the set of possible worlds in **S** that is compatible with some situation in Σ. Since implicit beliefs are defined with respect to a set of possible worlds, it is closed under implication and all valid sentences are implicitly believed. In this logic, a formula α is valid ($\models$) if it is true in all possible worlds w in any model $\mathbf{M_L}$. A sentence α is true or is *satisfied* at a situation s if $s \models_T \alpha$.

Explicit Belief : While implicit beliefs suffer from logical omniscience, explicit beliefs seem to have non-omniscient properties. In particular, the following set of sentences are satisfiable: $\mathbf{B}\alpha \wedge \mathbf{B}(\alpha \supset \beta) \wedge \neg\mathbf{B}\beta$ (explicit beliefs are not closed under implication); $\mathbf{B}\alpha \wedge \mathbf{B}(\neg\alpha) \wedge \neg\mathbf{B}\beta$ (agents can explicitly hold inconsistent beliefs); $\neg\mathbf{B}(\alpha \vee \neg\alpha)$ (valid sentences need not be explicitly believed); $\mathbf{B}\alpha \wedge \neg\mathbf{B}(\alpha \wedge (\beta \vee \neg\beta))$ (logically equivalent sentences need not be explicitly believed).

Nested Implicit and Explicit Belief : Levesque's logic do not allow nested beliefs. This issue has been taken up separately by Lakemeyer [13] and by Fagin and Halpern [5] in their logic of awareness (section 2.3). While, Levesque dispenses the use of accessibility relations (since he do not allow nested beliefs), Lakemeyer had employed not one but two accessibility relations. A Lakemeyer's model for implicit and explicit belief with a system of n agents* is a tuple $\mathbf{M_{LL}} = <\mathbf{S}, \mathbf{T}, \mathbf{F}, \mathbf{R}_1^+, \ldots, \mathbf{R}_n^+, \mathbf{R}_1^-, \ldots, \mathbf{R}_n^->$ where **S**, **T** and **F** are defined as above and $\mathbf{R}_i$+ (used to confirm a belief) and $\mathbf{R}_i$- (used to determine a disbelief) are binary relations on **S**. These two accessibility relations capture the intuition that in a given situation, an agent uses different sets of situations to confirm or disconfirm a belief. However, in a possible world, $\mathbf{R}_i$+ and $\mathbf{R}_i$- coincide, thus for all $w \in W(\Sigma)$, $w\ \mathbf{R}_i^+\ w_1 \Leftrightarrow w\ \mathbf{R}_i^- w_1$. A language **PL`** (similar to **PL**) is assumed, except that in **PL`** ,

* The model described here is a (direct) multi-agent extension of Lakemeyer's [13] model.

nesting of modal operators are allowed (however, no $\mathbf{L}_i$ may occur within the scope of a $\mathbf{B}_i$) and there are two sets of modal operators $\mathbf{L}_1,\ldots,\mathbf{L}_n$ and $\mathbf{B}_1,\ldots,\mathbf{B}_n$. In [13], the semantics for nested implicit and explicit beliefs are defined by replacing (L-TB), (L-FB), (L-TL) and (L-FL) with the following :

(LL-TB) $\mathbf{M}_{\mathbf{LL}},s \models_T \mathbf{B}_i\alpha$ iff for all $t \in \Sigma$ s $\mathbf{R}_i{+}\ t \Rightarrow \mathbf{M}_{\mathbf{LL}},t \models_T \alpha$

(LL-FB) $\mathbf{M}_{\mathbf{LL}},s \models_F \mathbf{B}_i\alpha$ iff for some $t \in \Sigma$ s $\mathbf{R}_i{-}\ t \wedge \mathbf{M}_{\mathbf{LL}},t \not\models_T \alpha$

(LL-TL) $\mathbf{M}_{\mathbf{LL}},s \models_T \mathbf{L}_i\alpha$ iff for all $t \in W(\Sigma)$ s $\mathbf{R}_i{+}\ t \Rightarrow \mathbf{M}_{\mathbf{LL}},t \models_T \alpha$

(LL-FL) $\mathbf{M}_{\mathbf{LL}},s \models_F \mathbf{L}_i\alpha$ iff $\mathbf{M}_{\mathbf{LL}}, s \not\models_T \mathbf{L}_i\alpha$.

In (**LL-TB**), $\mathbf{B}_i\alpha$ is true supported at *s* if all situations accessible from *s* via $\mathbf{R}_i+$ support the truth of α. For (**LL-FB**), the situation *s* supports the falsity of $\mathbf{B}_i\alpha$ just in case if there is a situation that is accessible from *s* via $\mathbf{R}_i-$ which do not support the truth of α.

2.2 Fagin-Halpern's Logic of Awareness

In their logic of awareness, Fagin and Halpern deal with both the issues of nested beliefs and multi-agent reasoning. A language **PL**, built from a set **P** of atomic letters {*p, p`, q, q`*,....} and the usual logical connectives $\neg$, $\wedge$, together with two sets of modal operators $\mathbf{L}_1,\ldots,\mathbf{L}_n$ and $\mathbf{B}_1,\ldots,\mathbf{B}_n$ is assumed. Thus, for $i = 1,\ldots,n$, $\mathbf{B}_i\alpha$ reads "agent *i* explicitly believe α" and $\mathbf{L}_i\alpha$ means "agent *i* implicitly believe α". Arbitrary nesting of the $\mathbf{L}_i$ and $\mathbf{B}_k$ in formulas are also permitted. Subsequently, a Kripke structure for awareness [5] is a tuple $\mathbf{M}_A = (\mathbf{S}, \mathbf{A}_1,\ldots,\mathbf{A}_n, R_1,\ldots,R_n, \pi)$ where **S** is a set of possible worlds, π is a truth assignment ($\pi : \mathbf{P} \to \{true, false\}$), $R_1,\ldots,R_n$ are serial, transitive and euclidean relations on **S** and $\mathbf{A}_i$ is a function associating each possible world with a set of primitive formulas $\mathbf{P} \supseteq \Psi$, where **P** is the set of all primitive sentences. That is, $\mathbf{A}_i(s)$ is the set of primitive formulas that agent *i* is aware of in state *s*. Two support relations $\models^{\Psi}_T$ and $\models^{\Psi}_F$ are defined for each set of formula Ψ. This places a restriction such that in every state only the formulas in Ψ are defined with either *true* or *false*. $\models^{\Psi}_T$ and $\models^{\Psi}_F$ together with the $\mathbf{A}_i(s)$ function provide some effect of partial situations from the perspective of agent *i*. For instance, a sentence α may be true at state *s*, but agent *i* may not be aware of α at *s* . Thus, for all $\psi \in$ **PL** the following semantics for the logic of awareness are defined [5] :

(A-*p*T) $\mathbf{M}_A,s \models^{\Psi}_T p$ iff $\pi(s, p) =$ true and $p \in \Psi$

(A-*p*F) $\mathbf{M}_A,s \models^{\Psi}_F p$ iff $\pi(s, p) =$ false and $p \in \Psi$

(A-*p*) $\mathbf{M}_A,s \models p$ iff $\pi(s, p) =$ true

(A-$\neg$T) $\mathbf{M}_A,s \models^{\Psi}_T -\alpha$ iff $\mathbf{M}_A,s \models^{\Psi}_F \alpha$

(A-$\neg$F) $\mathbf{M}_A,s \models^{\Psi}_F -\alpha$ iff $\mathbf{M}_A,s \models^{\Psi}_T \alpha$

(A-$\neg$) $\mathbf{M}_A,s \models -\alpha$ iff $\mathbf{M}_A,s \not\models \alpha$

(A-$\wedge$T) $\mathbf{M}_A,s \models^{\Psi}_T \alpha \wedge \beta$ iff $\mathbf{M}_A,s \models^{\Psi}_T \alpha$ and $\mathbf{M}_A,s \models^{\Psi}_T \beta$

(A-$\wedge$F) $\mathbf{M}_A,s \models^{\Psi}_F \alpha \wedge \beta$ iff $\mathbf{M}_A,s \models^{\Psi}_F \alpha$ or $\mathbf{M}_A,s \models^{\Psi}_F \beta$

(A-$\wedge$) $\mathbf{M}_A,s \models \alpha \wedge \beta$ iff $\mathbf{M}_A,s \models \alpha$ and $\mathbf{M}_A,s \models \beta$

(A-$\mathbf{B}_i$T) $\mathbf{M}_A,s \models^{\Psi}_T \mathbf{B}_i\alpha$ iff $\mathbf{M}_A,t \models_T^{\Psi \cap \mathbf{A}_i(s)} \alpha$ for all *t* such that $(s,t) \in R_i$

(A-$\mathbf{B}_i$F) $\mathbf{M}_A,s \models^{\Psi}_F \mathbf{B}_i\alpha$ iff $\mathbf{M}_A,t \models_F^{\Psi \cap \mathbf{A}_i(s)} \alpha$ for some *t* such that $(s,t) \in R_i$

(A-$\mathbf{B}_i$) $\mathbf{M}_A,s \models \mathbf{B}_i\alpha$ iff $\mathbf{M}_A,t \models_T^{\mathbf{P}} \mathbf{B}_i\alpha$, (**P** is the set of all primitive propositions)

(A-$\mathbf{L_i}$T) $\mathbf{M_A},s \models^{\Psi}{}_{T} \mathbf{L_i}\alpha$ iff $\mathbf{M_A},t \models^{\Psi}{}_{T}\alpha$ for all t such that $(s,t) \in R_i$

(A-$\mathbf{L_i}$F) $\mathbf{M_A},s \models^{\Psi}{}_{F} \mathbf{L_i}\alpha$ iff $\mathbf{M_A},t \models^{\Psi}{}_{F} \alpha$ for some t such that $(s,t) \in R_i$

(A-$\mathbf{L_i}$) $\mathbf{M_A},s \models \mathbf{L_i}\alpha$ iff $\mathbf{M_A},t \models \alpha$ for all t such that $(s,t) \in R_i$

Clause (A-*p*T) and (A-*p*F) give the property that for each set of primitive proposition Ψ : (i) $\mathbf{M_A},s \models^{\Psi}{}_{T} \alpha \Rightarrow \mathbf{M_A},s \models \alpha$ and (ii) $\mathbf{M_A},s \models^{\Psi}{}_{F} \alpha \Rightarrow \mathbf{M_A},s \models \neg\alpha$. Both (i) and (ii) are proven in Fagin and Halpern [5]. In clause (A-$\mathbf{B_i}$T), $\mathbf{M_A},s \models^{\Psi}{}_{T} \mathbf{B_i}\alpha$ means that agent i at state s *explicitly* believes α relative to Ψ, if α is true relative to $\Psi \cap \mathbf{A}_i(s)$ at all the states accessible from s. It also implies that $\mathbf{B_i}\alpha$ is true at s. This is because $\mathbf{M_A},s \models^{\Psi}{}_{T} \mathbf{B_i}\alpha \Rightarrow \mathbf{M_A},s \models \mathbf{B_i}\alpha$ follows from property (i) above. In the case of $\mathbf{M_A},s \models \mathbf{B_i}\alpha$, agent i explicitly believes α relative to **P**, where as for $\mathbf{M_A},s \models^{\Psi}{}_{T} \mathbf{B_i}\alpha$ agent i explicitly believes α relative to some subset of **P**. *Implicit* beliefs as defined in clause (A-$\mathbf{L_i}$) do not take awareness functions into account. Thus, $\mathbf{L_i}\alpha$ is true at s if α is true at all states accessible from s, regardless of whether agent i is aware of α.

3 Bilattice Theory

Bilattice theory due to Ginsberg [8] is a ramification of multi-valued logic. The motivating consideration of bilattice theory is to define formal structures that capture the bookkeeping information for several form of reasoning. It is based on the realization that a model of knowledge should capture two kinds of information : the degree of truth/ falsity and the level of certainty/ uncertainty.

Bilattice Definition : A bilattice is a set with two partial orderings $\leq_t$ and $\leq_k$. Given two truth values x and y, if $x \leq_t y$ then y is at least as true as x. The two operations corresponding to $\leq_t$ are the meet $\wedge$ and the joint $\vee$. For the k-partial ordering, if $x \leq_k y$ then y labels a sentence about which one has more knowledge than a sentence labeled with x. The two operations $\otimes$ and $\oplus$ respectively correspond to the meet and join in the k-ordering. Additionally, there is a unary negation operator that inverts the sense of the t-partial ordering while leaving the k -partial ordering unchanged. If $x \leq_t y$, then one will expect that $\neg y \leq_t \neg x$ and if $x \leq_k y$ then it follows that $\neg x \leq_k \neg y$. As in propositional logic, the negation operator in a bilattice also requires that $\neg\neg x = x$.

Definition 3.1 : A Bilattice is defined [8] as a sextuple $\{B, \wedge, \vee, \otimes, \oplus, \neg\}$ such that

(1) The t-lattice $\{B, \wedge, \vee\}$ and the k-lattice $\{B, \otimes, \oplus\}$ are both complete lattices.

(2) $\neg : B \rightarrow B$ is a mapping with :
(i) $\neg^2 = 1$, and
(ii) $\neg$ is a lattice homomorphism from $\{B, \wedge, \vee\}$ to $\{B, \vee, \wedge\}$ and $\{B, \otimes, \oplus\}$ to itself.

Ginsberg has proven that the bilattice framework subsumes several existing formalisms in knowledge representation. In [8], bilattices for first order logic (section 3.2), *default logic*, *prioritized default logic* and *assumption-based truth maintenance systems* were defined (space limitations preclude introducing all of the examples here).

A Bilattice for First Order logic : First order logic corresponds to a four point bilattice **B4** as shown in Figure 1. $\perp$ indicates no information is available and $\top$ indicates the presence of a contradiction. Negation corresponds to reflection around the k-axis (joining $\perp$ and $\top$). The four elements of the lattice shown in Figure 1 correspond to the maximal and minimal elements under the two partial orders. All bilattices will have these four distinguished elements.

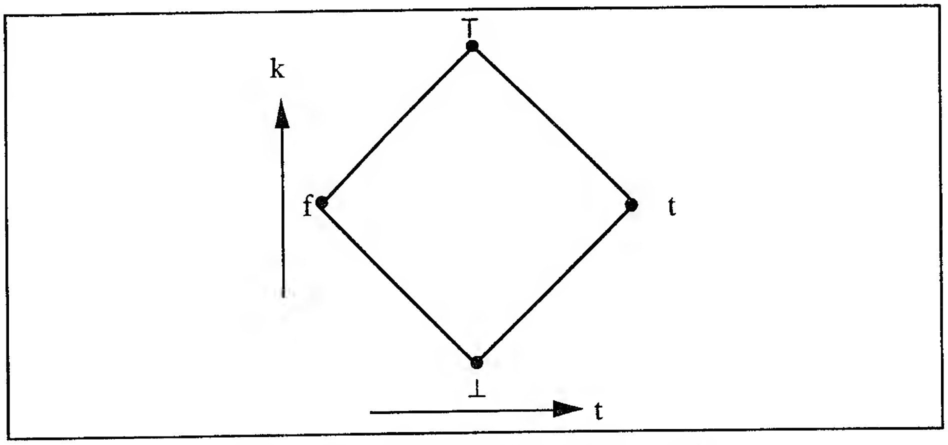


Fig. 1. $\mathbf{B_4}$**, The Smallest Non-trivial Bilattice**

4 Epistemic Operators and Truth Values

In this section, it is argued that various epistemic notions such as implicit belief, explicit belief and awareness can be associated with subsets of truth values in $\mathbf{B}_4$. This formalization is possible if one employs the *conflation operator* [6] to select situations with different degree of *incoherence* and *incompleteness.*

Definition 4.1 : Conflation (This is a an extension of Fitting's [6] conflation operator.)

Let $\mathbf{P}$ be a set of atomic letters and $\upsilon = <\upsilon^+, \upsilon^->$ be a valuation such that for all $p \in \mathbf{P}$, υ^+ maps p to $\{0,1\}$ and υ^- maps $\neg p$ to $\{0,1\}$, where $<0, 0>$, $<0, 1>$, $<1, 0>$ and $<1, 1>$. correspond to $\perp, f, t$ and $\top$ respectively. The conflation of a valuation υ is $- <\upsilon^+, \upsilon^-> = <- \upsilon^-, - \upsilon^+>$ where $- \upsilon^+$ and $- \upsilon^-$ are complements of υ^+ and υ^- respectively. Thus, for all $s \in \mathbf{S}$ and for all $p \in \mathbf{P}$,

(1) υ is exact iff $\upsilon(s)(p) = -\upsilon(s)(p)$ iff $\upsilon^+(s)(p) = -\upsilon^-(s)(p)$ and $\upsilon^-(s)(p) = -\upsilon^+(s)(p)$

(2) υ is consistent iff $\upsilon(s)(p) \leq_k -\upsilon(s)(p)$ iff $\upsilon^+(s)(p) \leq -\upsilon^-(s)(p)$ and $\upsilon^-(s)(p) \leq -\upsilon^+(s)(p)$

(3) υ is incoherent iff $\upsilon(s)(p) \geq_k -\upsilon(s)(p)$ iff $\upsilon^+(s)(p) \geq -\upsilon^-(s)(p)$ and $\upsilon^-(s)(p) \geq -\upsilon^+(s)(p)$

The conditions defined in definition 4.1 can be used for discerning situations with various degree of incoherence and incompleteness. The following properties of situations will be of interest : *exactness*, *consistency*, and *incoherence*. In addition the behavior of the conflation operation at each situation is defined in definition 4.3.

Definition 4.2 : exact, incomplete and incoherent situations :

(1) An *exact* situation is one which maps every $p \in \mathbf{P}$ to $\{<1,0>, <0,1>\}$.

(2) A *consistent* (incomplete or partial) situation is one which maps every $p \in \mathbf{P}$ to $\{<1,0>, <0,1>, <0, 0>\}$.

(3) An *incoherent* situation is one which maps every $p \in \mathbf{P}$ to $\{<1, 0>, <0,1>, <1, 1>\}$.

Definition 4.3 : Every situation s has a dual $\overline{s}$ such that

(1) for all $p \in \mathbf{P}$, $\upsilon(\overline{s})(p) = -\upsilon(s)(p)$.

(2) for all $t \in \mathbf{S}$, $t\,\mathbf{R}_i\,s$ <=> $\bar{t}\ \mathbf{R}_i\ \bar{s}$ where $\mathbf{R}_i$ is an accessibility relation.

(3) for all $w \in \mathbf{W}$, $w = \bar{w}$ where $\mathbf{W}$ is the set of all exact situations.

The constraints on situations defined in definition 4.1 to 4.3 provide the impetus for the formulation of a *multi-valued epistemic logic.*

4.1 Multi-valued Epistemic Logic (MEL)

A model for a multi-valued epistemic logic (**MEL**) with a system of n agents may be defined as $\mathbf{M} = <\mathbf{S}, \mathbf{R}_1,\ldots \mathbf{R}_n, \upsilon, ->$ where **S** is a set of situations and $\mathbf{R}_1,\ldots \mathbf{R}_n$ are binary relations. υ and $-$ are defined as in definition 4.1. A propositional language **PL** is also assumed. **PL** is formed with a set **P** of primitive letters $\{p_1,\ldots,p_n, q_1,\ldots,q_n, \ldots\}$ and standard connectives $\neg$, $\wedge$, $\vee$, $\supset$ and $\equiv$ together with the two sets of epistemic operators $\mathbf{L}_1,\ldots,\mathbf{L}_n$ and $\mathbf{B}_1,\ldots,\mathbf{B}_n$ (each $\mathbf{L}_i$ and each $\mathbf{B}_i$ are considered over exact situations and (arbitrary) situations respectively). In addition, there are two support relations ($\models_t$) and ($\models_f$) where $s \models_t \alpha$ means that s supports the truth of α and $s \models_f \alpha$ means that s supports the falsity of α. Thus, for all $\alpha, \beta \in$ **PL**, the semantics may be defined as follow :

(MEL-Tp) $\quad$ $\mathrm{M},s \models_t p \iff \upsilon^+(s)(p) = 1$

(MEL-Fp) $\quad$ $\mathrm{M},s \models_f p \iff \upsilon^-(s)(p) = 1$

(MEL-T$\neg$) $\quad$ $\mathrm{M},s \models_t \neg\alpha$ iff $\mathrm{M},s \models_f \alpha$

(MEL-F$\neg$) $\quad$ $\mathrm{M},s \models_f \neg\alpha$ iff $\mathrm{M},s \models_t \alpha$

(MEL-T$\wedge$) $\quad$ $\mathrm{M},s \models_t \alpha \wedge \beta$ iff $\mathrm{M},s \models_t \alpha$ and $\mathrm{M},s \models_t \beta$

(MEL-F$\wedge$) $\quad$ $\mathrm{M},s \models_f \alpha \wedge \beta$ iff $\mathrm{M},s \models_f \alpha$ or $\mathrm{M},s \models_f \beta$

(MEL-T$\mathbf{L}_i$) $\quad$ $\mathrm{M},s \models_t \mathbf{L}_i\,\alpha$ iff $\forall t \in \mathbf{W} s\mathbf{R}_i t \Rightarrow \mathrm{M},t \models_t \alpha$

(MEL-F$\mathbf{L}_i$) $\quad$ $\mathrm{M},s \models_f \mathbf{L}_i\,\alpha$ iff $\exists t \in \mathbf{W} s\mathbf{R}_i t \wedge \mathrm{M},t \models_f \alpha$

(MEL-T$\mathbf{B}_i$) $\quad$ $\mathrm{M},s \models_t \mathbf{B}_i\,\alpha$ iff $\forall t \in \mathrm{S}\ s\mathbf{R}_i t \Rightarrow \mathrm{M},t \models_t \alpha$

(MEL-F$\mathbf{B}_i$) $\quad$ $\mathrm{M},s \models_f \mathbf{B}_i\,\alpha$ iff $\exists\, t \in \mathrm{S}\ s\mathbf{R}_i t \wedge \mathrm{M},t \models_f \alpha$

In sections 4.2 and 4.3, the semantics presented above will be compared to those in the logic of implicit and explicit belief and the logic of awareness. However, to set the stage for the proofs in section 4.2, it seems prudent to state the following lemmas :

Lemma 4.1 : For all exact situations $w \in \mathbf{W}$ and for all $\psi \in$ **PL** the following hold:

(1) $\quad$ $\mathrm{M}, w \models_t \psi \iff \mathrm{M},w \not\models_f \psi$

(2) $\quad$ $\mathrm{M}, w \models_f \psi \iff \mathrm{M},w \not\models_t \psi$

Lemma 4.2 : For all situations $s \in \mathbf{S}$ and for all $\psi \in$ **PL** the following hold:

(1) $\quad$ $\mathrm{M}, \bar{s} \not\models_t \psi \iff \mathrm{M},s \models_f \psi$

(2) $\quad$ $\mathrm{M}, \bar{s} \not\models_f \psi \iff \mathrm{M},s \models_t \psi$

Proofs for Lemma 4.1 and Lemma 4.2 are by straightforward induction on the structure of ψ and are omitted.

4.2 On the Relations of MEL and the Logic of Implicit and Explicit Belief

In this section, it will be shown that for every model in the logic of implicit and explicit belief, there is a corresponding model in MEL.

Proposition 4.1 : Every model for the Levesque-Lakemeyer logic of implicit and explicit belief $\mathbf{M}_{\mathbf{LL}} = <\mathbf{S}, \mathbf{R}_1^+, \ldots \mathbf{R}_n^+, \mathbf{R}_1^-, \ldots \mathbf{R}_n^-, \mathbf{T}, \mathbf{F}>$ corresponds to a model in **MEL**, $\mathbf{M} = <\mathbf{S} \cup \overline{\mathbf{S}}, \mathbf{R}_1, \ldots \mathbf{R}_n, \upsilon, \rightarrow>$ preserving support of truth and falsity for all formulas $\psi \in$ **PL.**

Proof : Given a Levesque-Lakemeyer model $\mathbf{M}_{\mathbf{LL}} = <\mathbf{S}, \mathbf{R}_1^+, \ldots \mathbf{R}_n^+, \mathbf{R}_1^-, \ldots \mathbf{R}_n^-, \mathbf{T}, \mathbf{F}>$ defined a model in MEL, $\mathbf{M} = <\mathbf{S} \cup \overline{\mathbf{S}}, \mathbf{R}_1, \ldots \mathbf{R}_n, \upsilon, \rightarrow>$ as follows :

Let $\overline{\mathbf{S}} = \{ \bar{s} \mid s \in \mathbf{S}\}$ and $\mathbf{W} = \mathrm{W}(\Sigma)$ where $\mathrm{W}(\Sigma)$ is a set of complete situations in **S** and **W** is a set of exact situations in $\mathbf{S} \cup \overline{\mathbf{S}}$.

(AR) $\forall s, t \in \mathbf{S}\ s\mathbf{R}_i t \Leftrightarrow s\mathbf{R}_i^+ t$ and $s\mathbf{R}_i \bar{t} \Leftrightarrow s\mathbf{R}_i^- t$

(V) $\upsilon^+(s)(p) = 1 \Leftrightarrow s \in \mathbf{T}(p)$ and $\upsilon^-(s)(p) = 1 \Leftrightarrow s \in \mathbf{F}(p)$

To show by induction on the structure of ψ that for all $s \in \mathbf{S}$ the following hold :

(T) $\mathbf{M}_{\mathbf{LL}}, s \models_T \psi \Leftrightarrow \mathbf{M}, s \models_t \psi$

(F) $\mathbf{M}_{\mathbf{LL}}, s \models_F \psi \Leftrightarrow \mathbf{M}, s \models_f \psi$

(i) $\psi = p$ (Tp) $\mathbf{M}_{\mathbf{LL}}, s \models_T p \Leftrightarrow s \in \mathbf{T}(p)$ (L-Tp)

$\Leftrightarrow \upsilon^+(s)(p) = 1$ (V)

$\Leftrightarrow \mathbf{M}, s \models_t p$ (MEL-Tp)

The proof for (pF) is analogous to (pT) and is omitted. Also, the proof for the case when $\psi = \alpha \wedge \beta$ and $\psi = \neg\alpha$ follows easily by induction hypothesis.

(iv) $\psi = \mathbf{L}_i\, \alpha$ (LT) $\mathbf{M}_{\mathbf{LL}}, s \models_T \mathbf{L}_i\, \alpha \Leftrightarrow \forall t \in \mathrm{W}(\Sigma)\ s\mathbf{R}_i^+ t \Rightarrow \mathbf{M}_{\mathbf{LL}}, t \models_T \alpha$ (LL-TL$_i$)

$\Leftrightarrow \forall t \in \mathbf{W}\ s\mathbf{R}_i\, t \Rightarrow \mathrm{M}, t \models_t \alpha$ (AR & I.H.)

$\Leftrightarrow \mathrm{M}, s \models_t \mathbf{L}_i\, \alpha$ (MEL-TL$_i$)

(LF) $\mathbf{M}_{\mathbf{LL}}, s \models_F \mathbf{L}_i\, \alpha \Leftrightarrow \exists t \in \mathrm{W}(\Sigma)\ s\mathbf{R}_i^+ t \wedge \mathbf{M}_{\mathbf{LL}}, t \not\models_T \alpha$ (LL-FL$_i$)

$\Leftrightarrow \exists t \in \mathbf{W}\ s\mathbf{R}_i\, t \wedge \mathrm{M}, t \not\models_t \alpha$ (AR & I.H.)

$\Leftrightarrow \exists t \in \mathbf{W} s\mathbf{R}_i\, t \wedge \mathrm{M}, t \models_f \alpha$ (Lem. 4.1)

$\Leftrightarrow \mathrm{M}, s \models_f \mathbf{L}_i\, \alpha$ (MEL-FL$_i$)

(v) $\psi = \mathbf{B}_i\, \alpha$ (BT) $\mathbf{M}_{\mathbf{LL}}, s \models_T \mathbf{B}_i\, \alpha \Leftrightarrow \forall t \in \mathbf{S} s\mathbf{R}_i^+ t \Rightarrow \mathbf{M}_{\mathbf{LL}}, t \models_T \alpha$ (LL-TB$_i$)

$\Leftrightarrow \forall t \in \mathbf{S} s\mathbf{R}_i t \Rightarrow \mathrm{M}, t \models_t \alpha$ (AR & I.H)

$\Leftrightarrow \mathrm{M}, s \models_t \mathbf{B}_i\, \alpha$ (MEL-TB$_i$)

(BF) $\mathbf{M}_{\mathbf{LL}}, s \models_F \mathbf{B}_i\, \alpha \Leftrightarrow \exists t \in \mathbf{S} s\mathbf{R}_i^- t \wedge \mathbf{M}_{\mathbf{LL}}, t \not\models_T \alpha$ (LL-FB$_i$)

$\Leftrightarrow \exists\, \bar{t} \in \mathbf{S}\ \bar{s}\, \mathbf{R}_i\, \bar{t} \wedge \mathrm{M}, \bar{t} \not\models_t \alpha$ (AR & I.H.)

$\Leftrightarrow \exists t \in \mathbf{S}\ s\mathbf{R}_i\, t \wedge \mathbf{M}, t \models_f \alpha$ (Def. 4.3(2) & Lem. 4.2)

$\Leftrightarrow \mathbf{M}, s \models_f \mathbf{B}_i\, \alpha$ (MEL-FB$_i$)

4.3 On the Relations of MEL and the Logic of Awareness

In the logic of awareness, inconsistent situations are not allowed. Using the constraints defined in definition 4.2, it is argued that for every model in the logic of awareness, one can construct an equivalent model in **MEL** using only exact and consistent situations.

Proposition 4.2 : Every model for the logic of awareness $\mathbf{M}_A = <\mathbf{S}, \pi, R_1, \ldots R_n, \mathbf{A}_1, \ldots \mathbf{A}_n>$ corresponds to a model in **MEL**, $\mathbf{M} = <\mathbf{S}^{\backprime} \cup \mathbf{W}, \mathbf{R}_1, \ldots \mathbf{R}_n, \upsilon, ->$ preserving support of truth and falsity for all formulas $\psi \in \mathbf{PL}$.

Proof : Given a model for the logic of awareness $\mathbf{M}_A = <\mathbf{S}, \pi, R_1, \ldots R_n, \mathbf{A}_1, \ldots \mathbf{A}_n>$ define a model in **MEL**, $\mathbf{M} = <\mathbf{S}^{\backprime} \cup \mathbf{W}, \mathbf{R}_1, \ldots \mathbf{R}_n, \upsilon, ->$ as follows :

Let **P** be the set of all primitive atoms and Ψ and $\mathbf{A}_i(s)$ be a subsets of **P**. $\mathbf{S}^{\backprime}$ and **W** are sets of consistent situations and exact situations respectively.

(S) $\mathbf{S}^{\backprime} \cup \mathbf{W} = \mathbf{S} \times \wp(\mathbf{P})$ with $\mathbf{S}^{\backprime} = \{s^{\backprime} \mid s^{\backprime} = <s, \Psi \cap \mathbf{A}_i(s)> \text{ and } s \in \mathbf{S}\}$ and $\mathbf{W} = \{w \mid w = <s, \Psi> \text{ and } s \in \mathbf{S}\}$

(V) $\upsilon(s^{\backprime})(p) =$ $<1,0>$ iff $\pi(s)(p) = \text{true}$ and $p \in \Psi$
$<0,1>$ iff $\pi(s)(p) = \text{false}$ and $p \in \Psi$
$<0, 0>$ otherwise.

(AR1) $s^{\backprime}\ \mathbf{R}_i\ t^{\backprime} \iff s\ R_i\ t$ where $s^{\backprime} = <s, \Psi>$ and $t^{\backprime} = <t, \Psi \cap \mathbf{A}_i(s)>$

(AR2) $w\ \mathbf{R}_i\ w_1 \iff s\ R_i\ t$ where $w = <s, \Psi>$ and $w_1 = <t, \Psi>$

To show by induction on the structure of ψ that :

(T) $\mathbf{M}_A, s \models^{\Psi}_{T} \psi \iff \mathbf{M}, s^{\backprime} \models_t \psi$

(F) $\mathbf{M}_A, s \models^{\Psi}_{F} \psi \iff \mathbf{M}, s^{\backprime} \models_f \psi$

(i) $\psi = p$ (Tp) $\mathbf{M}_A, s \models^{\Psi}_{T} p \iff \pi(s)(p) = \text{true and } p \in \Psi$ (A-Tp)
$\iff \upsilon^{+}(s^{\backprime})(p) = 1$ (V)
$\iff \mathbf{M}, s^{\backprime} \models_t p$ (MEL-Tp)

The proof for (pF) is analogous to (pT) and is omitted. Also, the proof for the case when $\psi = \alpha \wedge \beta$ and $\psi = \neg\alpha$ follows easily by induction hypothesis.

(iv) $\psi = \mathbf{L}_i\alpha$ (LT) $\mathbf{M}_A, s \models^{\Psi}_{T} \mathbf{L}_i\alpha \iff \forall t \in \mathbf{S}\, sR_i t \Rightarrow \mathbf{M}_A, t \models^{\Psi}_{T} \alpha$ (A-T $\mathbf{L}_i$)
$\iff \forall t^{\backprime} \in \mathbf{W}\, s^{\backprime}\mathbf{R}_i t^{\backprime} \Rightarrow \mathbf{M}, t^{\backprime} \models_t \alpha$ (AR2 & I.H.)
$\iff \mathbf{M}, s^{\backprime} \models_t \mathbf{L}_i\alpha$ (MEL-T $\mathbf{L}_i$)

(LF) $\mathbf{M}_A, s \models^{\Psi}_{F} \mathbf{L}_i\alpha \iff \exists\, t \in \mathbf{S}\, sR_i t \wedge \mathbf{M}_A, t \models^{\Psi}_{F} \alpha$ (A-F$\mathbf{L}_i$)
$\iff \exists\, t^{\backprime} \in \mathbf{W}\, s^{\backprime}\mathbf{R}_i t^{\backprime} \wedge \mathbf{M}, t^{\backprime} \models_f \alpha$ (AR2 & I.H.)
$\iff \mathbf{M}, s^{\backprime} \models_f \mathbf{L}_i\alpha$ (MEL-F$\mathbf{L}_i$)

(v) $\psi = \mathbf{B}_i\alpha$ (BT) $\mathbf{M}_A, s \models^{\Psi}_{T} \mathbf{B}_i\alpha \iff \forall t \in \mathbf{S}\, sR_i t \Rightarrow \mathbf{M}_A, t \models_{T}^{\Psi \cap A(s)} \alpha$ (A-T $\mathbf{B}_i$)
$\iff \forall t^{\backprime} \in \mathbf{S}^{\backprime}\, s^{\backprime}\mathbf{R}_i t^{\backprime} \Rightarrow \mathbf{M}, t^{\backprime} \models_t \alpha$ (AR1 & I.H.)
$\iff \mathbf{M}, s^{\backprime} \models_t \mathbf{B}_i\alpha$ (MEL-T $\mathbf{B}_i$)

(BF) $\mathbf{M}_A, s \models^{\Psi}_{F} \mathbf{B}_i\alpha \iff \exists\, t \in \mathbf{S}\, sR_i t \wedge \mathbf{M}_A, t \models_{F}^{\Psi \cap A(s)} \alpha$ (A-F$\mathbf{B}_i$)
$\iff \exists t^{\backprime} \in \mathbf{S}^{\backprime}\, s^{\backprime}\mathbf{R}_i t^{\backprime} \wedge \mathbf{M}, t^{\backprime} \models_f \alpha$ (AR1 & I.H.)
$\iff \mathbf{M}, s^{\backprime} \models_f \mathbf{B}_i\alpha$ (MEL-F $\mathbf{B}_i$)

5 Conclusion and Further Work

In this paper evidence is provided for the claim that a uniform framework for reasoning about knowledge can be defined in the context of bilattice theory. The work reported here is only a preliminary step in the research. Further work will include establishing the relations of **MEL** and other epistemic logics; preliminary results are reported in [17]. It is hoped that this research can shed new light on both bilattice theory and epistemic logic.

Acknowledgments

Financial assistance for this work has been made available by the Department of Computer Science at the University of Calgary. I like to thank Brian Gaines and two anonymous referees for their insightful comments and suggestions for improving the final version of this paper.

References

[1] J. Barwise and J. Perry. Situations and Attitudes. Bradford Books, Cambridge, MA,1983.

[2] N. D. Jr. Belnap. A Useful Four-valued Logic. In Dunn and Epstein (Eds) Modern Uses of Multiple-valued Logic, pages 8-41. D. Reidel Publishing Company, 1977.

[3] B. Chellas. Modal Logic : An Introduction. Cambridge University Press, 1980.

[4] J. de Kleer. An Assumption-based Truth Maintenance System. Artificial Intelligence 28, pages 127-162, 1986.

[5] R. Fagin and J. Halpern. Belief, Awareness and Limited Reasoning. Artificial Intelligence, 34, pages 39-76, 1988.

[6] M. Fitting. Bilattices and the Semantics of Logic Programming. Journal of Logic Programming, volume 11, number 2, page 91-116, August, 1991.

[7] M. Ginsberg. Readings in Non-monotonic Reasoning. Morgan Kaufmann, Los Alto, CA, 1987.

[8] M. Ginsberg. Multivalued logics: A Uniform Approach to Reasoning in Artificial Intelligence. Computational intelligence 4, pages 265-316, 1988.

[9] J. Halpern and Y. Moses. A Guide to Completeness and Complexity for Modal Logics of Knowledge and Beliefs. Journal of Artificial Intelligence, volume 54, pages 319-379, 1992.

[10] J. Hintikka Knowledge and Belief. Cornell University Press, Ithaca, NY, 1962.

[11] J. Hintikka. Impossible possible worlds vindicated. Journal of Philosophical Logic, 4, pages 475-484, 1975.

[12] S. Kleen. Introduction of Metamathematics. Van Nostrand, 1952.

[13] G. Lakemeyer. Tractable Meta-reasoning in Propositional Logics of Beliefs. International Joint Conference in Artificial Intelligence, pages 402 - 408, 1987.

[14] H. Levesque. Logic of Implicit and Explicit Belief. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, pages 389-393, 1986.

[15] J. Lukasiewicz. “On 3-valued Logic”. In S. McCall (Eds) Polish Logic. Oxford University Press, 1967.

[16] N. Rescher. Many-valued Logic. McGraw Hill Book Company, 1969.

[17] K. M. Sim. A Multi-valued Epistemic Logic. Ph.D. dissertation. Knowledge Science Institute, University of Calgary, AB, CANADA, 1994 (to appear).

[18] H. Wangsing. A General Possible Worlds Framework for Reasoning About Knowledge and Belief. Studia Logical 49, pages 523 - 539, 1990.

Fast Termination of the Deductive Process in Resolution Proof Systems for Non-classical Logics

Zbigniew Stachniak *

Department of Computer Science, York University, Canada
email: zbigniew@cs.yorku.ca

Abstract. In this paper we discuss the problem of termination of the deductive process in resolution proof systems for non-classical logics.

1 Introduction

The deductive process of an automated reasoning program can be viewed as a process of generation and accumulation of information (syntactic or semantic) and testing it against the class of termination conditions specified in advance. These termination conditions specify whether the deductive process should terminate (successfully or unsuccessfully) or whether it should continue. Usually, a reasoning program generates and retains too much information in its database; it often generates facts that are already in the database or are irrelevant for the successful termination of a reasoning task. One way of gaining efficiency of the deductive process is to develop theorem proving strategies for restricting and directing the application of inference rules. These speedup techniques force certain choices of deduction steps (and block other) so that the termination conditions can be satisfied as quickly as possible (cf. [3,4,9,11]). Another way of gaining efficiency of a reasoning program is a careful selection of such termination conditions, which for their satisfaction, would require as little information to be generated as possible. In clausal resolution proof systems for the classical first-order logic, the termination condition coincides with the empty clause detection. This termination condition is sufficiently simple and, hence, most of the work done on improving efficiency of the reasoning process concentrates on theorem proving strategies and efficient search techniques (cf. [1,4,11]). In analytic tableau theorem provers, the termination condition is satisfied when all the branches of a tableau are completely developed (as closed or open). This termination condition can be improved, for example, by detecting some of the branches that will eventually become closed without actually closing them (cf. [10]).

In non-clausal resolution proof systems for non-classical logics (cf. [5,6,7,9]), the successful termination of a deductive process is indicated by deducing one

* Research supported by the grant from the Natural Sciences and Engineering Research Council of Canada

of the preselected finite sets of formulas. These sets, called the *terminal sets*, are inconsistent and form an integral part of the description of a resolution proof system. If Rs is a resolution proof system and $\mathcal{F}$ is such a family of terminal sets, then the termination of the deductive process is specified as follows: given a finite set X of formulas,

(T1) *X is refutable in Rs iff a set $V \in \mathcal{F}$ can be deduced from X using the inference rules of Rs.*

As every set in $\mathcal{F}$ is inconsistent, the deductive process terminates successfully as soon as an explicit contradiction (represented by any of the sets in $\mathcal{F}$) is deduced from X. However, the family $\mathcal{F}$, as well as the elements of $\mathcal{F}$, can be large. Hence, the verification of the right hand side of (T1) may require a substantial number of the refutation steps. Our experimentation with a number of resolution proof systems has shown that for many of them the termination condition (T1) can be replaced by a simpler test: *there exists a finite set V of formulas such that*

(T2) *X is refutable in Rs iff a formula from V can be deduced from X using the inference rules of Rs.*

In view of (T2), a refutational process terminates successfully as soon as one of the formulas of V is deduced from X. In comparison with (T1), a successful termination of the deductive process is 'witnessed' by a single formula (i.e. an element of V) rather than a set of formulas (i.e. an element of $\mathcal{F}$). As a result, we can save a substantial number of deductive steps in the search for a refutation.

Among the logics satisfying (T2) there is the classical propositional logic, finitely-valued logics of Łukasiewicz and Post, three-valued logics of Heyting, Kleene, Słupecki, Sobociński, etc. (cf. [2]), as well as some non-monotonic inference systems with the consistency preservation property (cf. [8]). There are, however, logical calculi and their resolution proof systems for which (T1) cannot be replaced by (T2). Therefore, determining the availability of the termination test (T2) is important for those reasoning programs which are to be implemented as non-clausal resolution proof systems. In this paper we identify one class of logics for which resolution proof systems satisfying (T2) can be constructed. For this class of logics, we discuss the effect of the replacement of (T1) by (T2) on the reduction of the length of a refutation.

This paper discusses only propositional logics and it is organized as follows. In Sections 2 and 4 we briefly review the notions of a logical system, logical matrix, and of a resolution proof system for a logical calculus. In Section 3, we use the example of the three-valued Łukasiewicz logic and one of its resolution proof systems to discuss the termination condition (T1) and the effect of its replacement by (T2) on the search for a refutation. Section 5 brings the main result of this paper – the description of a certain class of logical calculi with the termination test (T2). For the logics in this class, we give an effective way to construct a set V that satisfies (T2). We conclude the paper by revisiting the example discussed in Section 3 and applying the results of Section 5 to the formal construction of the termination test (T2) for the three-valued Łukasiewicz logic and its minimal resolution proof system.

2 Logical Preliminaries

In this section we briefly review the notions of a propositional logic and of a logical matrix. For more complete treatment of the material of this section, one can consult [12].

By a *propositional logic* we understand a pair $\mathcal{P} = \langle \mathcal{L}, C \rangle$, where $\mathcal{L}$ is a propositional language and C is a consequence operation on $\mathcal{L}$. A *propositional language* is specified by a finite collection $f_0, ..., f_k$ of logical connectives and by a countable infinite set $\{p_0, p_1, ...\}$ of propositional variables. The set L of well-formed formulas of a language $\mathcal{L}$ is defined using the usual formula-formation rules. The second element in the definition of a logic, the *consequence operation* C, is a function mapping 2^L into 2^L and satisfying the following two conditions:

(c1) for every $X \subseteq L$, $X \subseteq C(X) = C(C(X))$;
(c2) for every $X, Y \subseteq L$, $X \subseteq Y$ implies $C(X) \subseteq C(Y)$.

If X is a set of formulas and α is a formula, then the expression '$\alpha \in C(X)$' can be read as 'α is a consequence of X' or as 'α is provable from X'. Hence, we view a logic not as a set of formulas closed under some collection of inference rules, but as a language together with a provability relation. We call a set $X \subseteq L$ *consistent* (resp. *inconsistent*), if $C(X) \neq L$ (resp. $C(X) = L$). If α and v are formulas and p is a propositional variable, then we shall write $\alpha(p/v)$ to denote the result of simultaneous replacement of every occurrence of p in α by v.

A logic $\mathcal{P}$ is said to be *standard* if, in addition to (c1) and (c2), the following conditions hold: for every $X \subseteq L$ and every $\alpha \in L$,

(c3) if $\alpha \in C(X)$, then $e(\alpha) \in C(e(X))$, for every logical substitution e;
(c4) $C(X) = \bigcup \{C(X_f) : X_f \subseteq X \textit{ and } X_f \textit{ is finite}\}$;
(c5) $C(\emptyset) \neq L$ and $C(Y) = L$, for some finite $Y \subseteq L$.

While (c1) and (c2) describe the general properties of a consequence operation (*inclusion, idempotence*, and *monotonicity*), (c3) (*structurality*) and (c4) (*compactness*) are satisfied by most (but not all) calculi studied in formal logic or computer science. By (c5), the set $C(\emptyset)$ of tautologies is consistent and there exists at least one finite inconsistent set of formulas. If a logic $\mathcal{P}$ does not satisfy (c5), then refutational methods for $\mathcal{P}$ are either trivial (if $C(\emptyset) = L$, then, by (c2), every set of formulas is inconsistent in this logic) or are not available at all (if $C(Y) \neq L$, for every finite $Y \subseteq L$, then there are no finite sets of formulas that can be refuted). From now on, we shall concentrate on standard logics exclusively. Unless stated otherwise, the term 'logic' means a standard logic.

Propositional logics can be semantically defined using logical matrices. Let $\mathcal{L}$ be a propositional language with the connectives $f_0, ..., f_k$. A logical matrix for $\mathcal{L}$ is a system of the form

$$M = \langle \langle A, F_0, ..., F_k \rangle, D \rangle,$$

where A is a non-empty set (of ***truth-values***), $F_0, ..., F_k$ are operations on A, and D is a subset of A. We assume that for every $i \leq k, f_i$ and F_i are of the same arity. The operation F_i, $i \leq k$, is the truth-functional interpretation of the connective f_i. Finally, the set D consists of ***designated truth-values*** of M. Every mapping h^* from the set of propositional variables of $\mathcal{L}$ into A can be uniquely extended into the mapping h of L into A in an obvious way. We call h a ***logical valuation*** of $\mathcal{L}$ into M. Every logical matrix M on $\mathcal{L}$ defines the consequence operation C_M in the following way: for every $X \cup \{\alpha\} \subseteq L$,

$$\alpha \in C_M(X) \textit{ iff } \text{for every logical valuation } h, \text{ if } h(X) \subseteq D, \text{ then } h(\alpha) \in D.$$

The logic $\langle \mathcal{L}, C_M \rangle$ is always structural but not necessarily standard.

3 Termination Test for 3-Valued Łukasiewicz Logic

We begin our search for an efficient resolution termination test by examining the three-valued Łukasiewicz logic $\mathcal{P}_3$ and one of its resolution proof systems. With this example we demonstrate the need for an efficient termination test; we propose a simple termination test, and discuss the effect of our proposal on the search for a refutation. In Section 5, we shall generalize our findings to other non-classical logics.

The logic $\mathcal{P}_3$ is defined by the three-valued logical matrix $M_3 = \langle \mathcal{A}, \{2\} \rangle$, where $\mathcal{A} = \langle \{0, 1, 2\}, \underline{\vee}, \underline{\wedge}, \underline{\rightarrow}, \underline{\neg} \rangle$ is the algebra of truth-values of M_3, and 2 is the only designated truth-value. The operations $\underline{\vee}, \underline{\wedge}, \underline{\rightarrow}$ and $\underline{\neg}$ are defined by the well-known truth-tables:

$\underline{\vee}$	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

$\underline{\wedge}$	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

$\underline{\rightarrow}$	0	1	2
0	2	2	2
1	1	2	2
2	0	1	2

	$\underline{\neg}$
0	2
1	1
2	0

In [5] the following non-clausal resolution proof system Rs_3 for this logic is given. Rs_3 has two types of inference rules: the resolution rule and the transformation (or simplification) rules. The *resolution rule* has the form:

$$\frac{\alpha_0(q), \alpha_1(q), \alpha_2(q)}{\alpha_0(q/v_3) \vee \alpha_1(q/v_4) \vee \alpha_2(q/v_5)}$$

where $\alpha_0(q), \alpha_1(q), \alpha_2(q)$ are arbitrary formulas sharing a variable q, and v_3, v_4, and v_5 are preselected formulas:

$$v_5 = p_0 \rightarrow p_0, \quad v_4 = p_0 \vee \neg p_0, \quad v_3 = \neg(v_4 \rightarrow \neg v_4).$$

The formula $\alpha_0(q/v_3) \vee \alpha_1(q/v_4) \vee \alpha_2(q/v_5)$, called the *resolvent* of $\alpha_0(q), \alpha_1(q)$, and $\alpha_2(q)$, is obtained by replacing every occurrence of the variable q in α_0 by v_3, every occurrence of q in α_1 by v_4, and every occurrence of q in α_2 by v_5, and, finally, by taking the disjunction of the results. The second type of the inference rules of Rs_3 are the *transformation rules*, i.e., expressions of the form:

$$(v_i \vee v_j) \Rightarrow v, \ \ (v_i \wedge v_j) \Rightarrow v, \ \ (v_i \rightarrow v_j) \Rightarrow v, \ \ \neg(v_i) = v,$$

where v, v_i, v_j are either the formulas v_3, v_4, v_5 or their negations:

$$v_2 = \neg v_3, \ \ v_1 = \neg v_4, \ \ v_0 = \neg v_5.$$

We call the set $Ver = \{v_0, v_1, v_2, v_3, v_4, v_5\}$ the set of *verifiers* of Rs_3. The role of the transformation rules is to simplify formulas generated during the deductive process. In a single application of a transformation rule $w \Rightarrow v$ to a formula α, we rewrite every occurrence of w in α as v. All the transformation rules of Rs_3 can be conveniently represented by the following tables:

→	0	1	2	3	4	5
0	5	5	5	5	5	5
1	4	5	5	4	5	5
2	3	4	5	3	4	5
3	2	2	2	5	5	5
4	1	2	2	4	5	5
5	0	1	2	3	4	5

∨	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	1	2	4	4	5
2	2	2	2	5	5	5
3	3	4	5	3	4	5
4	4	4	5	4	4	5
5	5	5	5	5	5	5

∧	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	1	0	1	1
2	0	1	2	0	1	2
3	0	0	0	3	3	3
4	0	1	1	3	4	4
5	0	1	2	3	4	5

	¬
0	5
1	4
2	3
3	2
4	1
5	0

The rows, columns, and values recorded in these tables are the indices of the verifiers of Rs_3. Hence, according to the table representing the transformation rules for $\wedge$, $v_0 \wedge v_1$ can be rewritten as v_0, while $v_4 \wedge v_1$ as v_1.

Let $X = \{p_1, \ p_1 \rightarrow p_2, \ p_2 \rightarrow p_3, \ \neg p_3\}$. This set is inconsistent in $\mathcal{P}_3$. In order to refute X in Rs_3, we shall use the inference rules of Rs_3 to deduce new formulas from X. We shall terminate (successfully) this deductive process as soon as an inconsistent subset of Ver is generated. The inconsistent sets of verifiers of Rs_3 form a part of the description of Rs_3 and should be computed in advance. Hence, the termination test for Rs_3 can be informally expressed by the following equivalence:

(T1*) *a set Y of formulas is refutable in Rs_3 iff a set $V \in \mathcal{F}$ can be deduced from Y using the inference rules of Rs_3,*

where $\mathcal{F}$ is the family of all inconsistent sets of verifiers of Rs_3. We assume that the variable p_0, used to construct the verifiers of Rs_3, does not occur in the formulas of Y. We can begin the refutation of X by applying the resolution rule to the sequence $p_2 \rightarrow p_3, \ \ p_2 \rightarrow p_3, \ \ \neg p_3$ and obtain the resolvent

(1) $(p_2 \rightarrow v_3) \vee (p_2 \rightarrow v_4) \vee \neg v_5,$

which reduces to the formula

(2) $\gamma(p_2) = (p_2 \rightarrow v_3) \vee (p_2 \rightarrow v_4) \vee v_0.$

Looking at the 0-th column in the table listing the transformation rules for $\vee$, we conclude that no matter what verifiers will eventually replace the variable p_2 in (2), the role of v_0 in (2) is redundant and, hence, we can rewrite (2) as

(3) $\gamma(p_2) = (p_2 \to v_3) \vee (p_2 \to v_4)$.

Next, we resolve $p_1 \to p_2$, $p_1 \to p_2$, and $\gamma(p_2)$ upon p_2 and obtain

(4) $(p_1 \to v_3) \vee (p_1 \to v_4) \vee \gamma(p_2/v_5)$,

which reduces to

(5) $\delta(p_1) = (p_1 \to v_3) \vee (p_1 \to v_4) \vee v_4$.

Finally, we apply the resolution rule to the sequence p_1, p_1, $\delta(p_1)$ and obtain

(6) $v_3 \vee v_4 \vee \delta(p_1/v_5)$,

which reduces to

(7) v_4.

As $\{v_4\} \notin \mathcal{F}$, the deductive process cannot be terminated at this point. What we can do is to resolve $\neg p_3, \neg p_3, \neg p_3$ to obtain

(8) $\neg v_3 \vee \neg v_4 \vee \neg v_5$,

which reduces to

(9) v_2.

Now, the deduced formulas (7) and (9), i.e., the verifiers v_4 and v_2, form an inconsistent set, and, by (T1*), the refutation of X terminates successfully.

From the refutation process just described we conclude that the right hand side of the test (T1) should be checked every time a new verifier is being added to the set of deduced formulas. With every deduction of a verifier, we should search the family $\mathcal{F}$ to see if the verifiers deduced so far form a *terminal set*, i.e., an element of $\mathcal{F}$. As the terminal sets can be large, a substantial number of deductive steps may be required to deduce one of them. Another difficulty with the implementation of (T1) is caused by the fact that the family $\mathcal{F}$ of terminal sets can be quite large (in our example, $\mathcal{F}$ consists of 54 elements). This may require the employment of special data structures to represent and search $\mathcal{F}$ in order to determine whether or not the deductive process should be terminated.

It turns out that for the Łukasiewicz logic $\mathcal{P}_3$ and for its resolution counterpart Rs_3 a simpler termination test can be proposed:

(T2*) *the deductive process terminates successfully iff a verifier other than v_2 and v_5 can be deduced.*

The correctness of this test will be discussed at the end of Section 5. In the light of (T2*), we may consider all the verifiers other than v_2 and v_5 to be 'witnesses' of inconsistency. As soon as one of such witnesses is deduced, we have sufficient information to successfully terminate the deductive process. This makes not only the storage of the family $\mathcal{F}$, and the search through it, unnecessary, but it also significantly speeds up the refutation process, as the successful refutation is witnessed now by a single verifier rather than a set of verifiers. By reexamining our refutation of the set X, we conclude that the deductive process can be terminated after the deduction of v_4 in step (7). Indeed, by (T2*) we have already generated enough information to terminate the deductive process. The generation of the verifier v_2 is now redundant and a number of deductive steps have been saved. Curiously enough, the verifier v_4, which we have used to terminate the refutation of X, is not a contradictory formula, as it is satisfied by any valuation h such that $h(p_0) = 2$. This means that for a successful termination of the deductive process we don't necessarily need to deduce an explicit contradiction (represented by unsatisfiable verifier v_0 or v_1); as far as (T2*) is concerned, the deduction of the satisfiable verifier v_3 or v_4 is equally sufficient.

4 Resolution Logics

In this section we briefly review the notions of a resolution proof system and of a resolution logic (cf. [6,7,9]) required to extend the scope of our discussion on the termination test (T2), initiated in Section 3, to other non-classical logics.

Let $\mathcal{L}$ be an arbitrary propositional language fixed for the rest of this section. We assume that $f_0, ..., f_k$ are all the connectives of $\mathcal{L}$ and that among them there is a binary connective which we intend to interpret as a disjunction (we shall use the symbol '$\vee$' to denote this connective). A *resolution proof system Rs* on $\mathcal{L}$ is defined by specifying:

- a finite and non-empty set Ver of formulas of $\mathcal{L}$ called the *verifiers* of Rs,
- a list $\underline{f_0}, ..., \underline{f_k}$ of operations on Ver; we assume that for every $i \leq k$, the connective f_i and the operation $\underline{f_i}$ are of the same arity;
- a non-empty family $\mathcal{F}$ of subsets of Ver.

The system Rs_3 defined in Section 3 can serve as an example of a resolution proof system. With Rs, we associate two types of inference rules: the resolution rule and the transformation rules. Since these rules are fully discussed elsewhere (cf. [6,7,9]) we only quote their respective definitions. The *resolution rule* is the set of all sequents:

$$\frac{\alpha_0(p), ..., \alpha_m(p)}{\alpha_0(p/v_0) \vee ... \vee \alpha_m(p/v_m)},$$

where $\alpha_0, ..., \alpha_m$ are arbitrary formulas of $\mathcal{L}$ with a common variable p, and $v_0, ..., v_m$ is a fixed enumeration of the set Ver of verifiers. We assume that p does not occur in the verifiers of Rs. Let f_i be a k-ary connective and let $w, w_1, ..., w_k$ be verifiers of Rs. An expression

$$f_i(w_1, ..., w_k) \Rightarrow w$$

is a *transformation rule* of Rs, provided that $\underline{f_i}$ applied to the verifiers $w_1, ..., w_k$ yields w, i.e., when $\underline{f_i}(w_1, ..., w_k) = w$. Hence, the role of the operations $\underline{f_0}, ..., \underline{f_k}$ is to describe the complete set of the transformation rules.

Let $Rs = \langle \mathcal{V}, \mathcal{F} \rangle$ be a resolution proof system on $\mathcal{L}$. The set Ver of verifiers together with the operations $\underline{f_0}, ..., \underline{f_k}$ are used to define the inference rules in terms of which the deductive process is carried out. The role of the family $\mathcal{F}$ is to provide the termination conditions for such process. The deductive process can be viewed as the execution of the following steps. Let X be a finite set of formulas that do not share variables with the verifiers of Rs. Henceforth, we shall call such sets *clean* in Rs.

(A) Generate the set $\mathcal{T}(X)$ of all resolvents of formulas of X. If $\mathcal{T}(X) \subseteq X$, then the deductive process terminates unsuccessfully. Else, let $X := X \cup \mathcal{T}(X)$.

(B) Let Z be the result of simplification of formulas of X using the transformation rules of Rs. If for some $V \subseteq Z, V \in \mathcal{F}$, then the deductive process terminates successfully. Else return to step (A).

We say that 'X is refutable in Rs', if the deductive process described above terminates successfully on input X. In Section 3 we have described a refutation of a set of formulas of the three-valued Lukasiewicz logic. Let us note, however, that this refutation is obtained using a 'simplified' form of the resolution rule.

Let $\mathcal{P} = \langle \mathcal{L}, C \rangle$ be a propositional logic. Let $\Theta_{\mathcal{P}}$ be an equivalence relation on the set L of all formulas of $\mathcal{L}$ defined as follows: for every $\alpha, \beta \in L$,

$\alpha \Theta_{\mathcal{P}} \beta$ *iff* for every set $X \subseteq L$ and every formula $\gamma(p) \in L, X \cup \{\gamma(p/\alpha)\}$ is inconsistent in $\mathcal{P}$ iff $X \cup \{\gamma(p/\beta)\}$ is inconsistent in $\mathcal{P}$.

In other words, two formulas are $\Theta_{\mathcal{P}}$-equivalent if and only if the replacement of α by β does not change the inconsistency status of any context in which such a replacement may be performed. We call a resolution proof system Rs on $\mathcal{L}$ a *resolution counterpart* of $\mathcal{P}$ (and $\mathcal{P}$ a *resolution logic*) provided that:

(r1) if $X \subseteq L$ is finite and clean in Rs, then X is inconsistent in $\mathcal{P}$ *iff* X is refutable in Rs;

(r2) for every transformation rule $f(w_1, ..., w_k) \Rightarrow w$, $f(w_1, ..., w_k)\Theta_{\mathcal{P}} w$;

(r3) the family $\mathcal{F}$ of Rs consists of all subsets of the set of verifiers of Rs which are inconsistent in $\mathcal{P}$.

Condition (r1) describes the *refutational completeness* of $\mathcal{P}$ with respect to Rs. By (r2), the application of a transformation rule does not affect the consistency of a context in which the simplification is done, while by (r3), the terminal sets of Rs coincide with inconsistent sets of verifiers of Rs. The class of resolution logics is characterized in [6].

5 Simplification of the Termination Test

In this section we describe a certain class of resolution logics for which the termination test (T1) can be replaced by (T2).

Let $\mathcal{L}$ be a propositional language with at least one binary connective $\vee$ and let $\mathcal{C}$ be the class of all resolution logics $\mathcal{P} = \langle \mathcal{L}, C \rangle$ for which there exists a finite matrix $M = \langle \mathcal{A}, D \rangle$ (i.e. a matrix with a finite number of truth-values) such that:

(s1) C and the consequence operation C_M defined by M have the same inconsistent sets;
(s2) if v, w are arbitrary truth-values of M, then $v \vee w \in D$ *iff* $v \in D$ or $w \in D$;
(s3) for every $\alpha, \beta \in L$, $\alpha \Theta_{\mathcal{P}} \beta$ *iff* for every valuation h of $\mathcal{L}$ into $M, h(\alpha) = h(\beta)$.

The relation $\Theta_{\mathcal{P}}$ has been defined in the previous section. Let us briefly comment on (s1), (s2), and (s3). Condition (s1) says that, from a refutational point of view, $\mathcal{P}$ is a finitely-valued logic. Condition (s2) guarantees that the logic $\langle \mathcal{L}, C_M \rangle$ is disjunctive (i.e., that $\vee$ is a disjunction of $\langle \mathcal{L}, C_M \rangle$, cf. [12]). Finally, (s3) provides a simple criterion for testing $\Theta_{\mathcal{P}}$-equivalence of formulas. The class $\mathcal{C}$ contains, among other propositional calculi, the classical logic and the intuitionistic logic, logical systems intermediate between the intuitionistic and the classical logic, logical calculi having the same inconsistent sets of formulas as the finitely-valued logics of Łukasiewicz or Post, the three-valued logics of Heyting, Słupecki, and Sobociński. The main theorem of this paper, stated below, shows that the conditions (s1), (s2), and (s3) imply the existence of the test (T2), for every resolution counterpart of $\mathcal{P}$.

THEOREM 1: Let $\mathcal{P} \in \mathcal{C}$ and let Rs be a resolution counterpart of $\mathcal{P}$. Then, there exist: a set V of verifiers of Rs and verifiers $v_0, \ldots, v_{n-1}$ of Rs, $n \geq 2$, such that for every finite $X \subseteq L$ clean in Rs,

(T2) X is inconsistent in $\mathcal{P}$ *iff* a verifier $v \in V$ can be deduced from X using the transformation rules of Rs and the following simplified form of the resolution rule:

$$R_n : \quad \frac{\alpha_0(p), \ldots, \alpha_{n-1}(p)}{\alpha_0(p/v_0) \vee \ldots \vee \alpha_{n-1}(p/v_{n-1})}.$$

Moreover, V and the verifiers $v_0, \ldots, v_{n-1}$ can be selected effectively.

Let us point out that Theorem 1 provides only the sufficient conditions for the existence of (T2) for every resolution counterpart of a logical calculus.

Sketch of the proof: Let $\mathcal{P}$ and Rs be as stated and let $M = \langle \mathcal{A}, D \rangle$ be a matrix which, together with $\mathcal{P}$, satisfies (s1), (s2), and (s3). Let us select a valuation h of $\mathcal{L}$ into M such that:

(a) h maps the verifiers of Rs onto the set of truth-values of M;
(b) the set d of all verifiers of Rs that are mapped by h into D is maximal (with respect to $\subseteq$) among C_M-consistent subsets of Ver.

The proof of the existence of such a valuation can be found in [9]. Let $V = Ver - d$ and let $v_0, \ldots, v_{n-1}$ be any selection of verifiers of Rs that are mapped by h onto the set of truth-values of M. By (a) and by Lemma 3.0 from [5], the resolution rule of Rs can be simplified to the required form. The proof that the set V has been chosen correctly is left to the reader. □

Resolution counterparts of logics in $\mathcal{C}$ can be constructed effectively. For example, we can use the resolution proof system generation algorithm presented in [6]. In the light of Theorem 1, the resolution rule of these resolution proof systems can be simplified and used with the termination condition (T2).

The three-valued Lukasiewicz logic $\mathcal{P}_3$ belongs to the class $\mathcal{C}$. Indeed, $\mathcal{P}_3$ and the matrix M_3 (described in Section 3) satisfy (s1), (s2), and (s3). Hence, by Theorem 1, there exists a termination test (T2) for $\mathcal{P}_3$ and every resolution proof system for this calculus. To derive the termination condition (T2*) for $\mathcal{P}_3$ and its resolution counterpart Rs_3 (see Section 3), let h be any valuation of $\mathcal{L}$ into M_3 such that $h(p_0) = 1$ and let $d = \{v_2, v_5\}$. It can be easily verified that h and d satisfy the conditions (a) and (b) stated in the proof of Theorem 1. Moreover, since $h(\{v_3, v_4, v_5\}) = \{0, 1, 2\}$, the resolution rule of Rs_3 can be simplified to the form

$$\frac{\alpha_0(q), \alpha_1(q), \alpha_2(q)}{\alpha_0(q/v_3) \vee \alpha_1(q/v_4) \vee \alpha_2(q/v_5)}.$$

This justifies why in the refutation presented in Section 3 we have used this form of the resolution rule rather than its general form described in Section 4.

References

1. Benanav, D. (1992). Recognizing Unnecessary Clauses in Resolution Based Systems. *Journal of Automated Reasoning* **9**, 43–76.
2. Bolc, L. and Borowik, P. (1992). *Many-Valued Logics 1. Theoretical Foundations.* Springer-Verlag.
3. Chang, C. and Lee, R. (1973). *Symbolic Logic and Mechanical Theorem Proving,* Academic Press.
4. Eisinger, N. and Ohlbach, H.J. (1993). Deductive Systems Based on Resolution. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson (editors), *Handbook of Logic in Artificial Intelligence and Logic Programming,* vol. 1, Clarendon Press.
5. Harley, E. and Stachniak, Z. (1993). Minimal Resolution Proof Systems for Finitely-Valued Lukasiewicz Logics, in *Proceedings of the Twenty-Third International Symposium on Multiple-Valued Logic,* Sacramento, 54-59.
6. Stachniak, Z. (1991). Extending Resolution to Resolution Logics. *Journal of Experimental and Theoretical Artificial Intelligence* **3**, 17–32.
7. Stachniak, Z. (1993). An Essay on Resolution Logics. *Studia Logica* **52**, 309–322.
8. Stachniak, Z. (1993). Algebraic Semantics for Cumulative Inference Operations, in *Proceedings of the AAAI'93,* Washington, DC, 444–449. AAAI Press/MIT Press.
9. Stachniak, Z. (1994). *Resolution Proof Systems: An Algebraic Theory,* to be published.
10. Vellino, A. (1993). The Relative Complexity of Analytic Tableaux and SL-Resolution. *Studia Logica* **52**, 323–337.
11. Wos, L. (1988). *Automated Reasoning: 33 Basic Research Problems,* Prentice Hall.
12. Wójcicki, R. (1988). *Theory of Logical Calculi: Basic Theory of Consequence Operations.* Kluwer Academic Publishers.

Springer-Verlag and the Environment

We at Springer-Verlag firmly believe that an international science publisher has a special obligation to the environment, and our corporate policies consistently reflect this conviction.

We also expect our business partners – paper mills, printers, packaging manufacturers, etc. – to commit themselves to using environmentally friendly materials and production processes.

The paper in this book is made from low- or no-chlorine pulp and is acid free, in conformance with international standards for paper permanency.

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 679: C. Fermüller, A. Leitsch, T. Tammet, N. Zamov, Resolution Methods for the Decision Problem. VIII, 205 pages. 1993.

Vol. 681: H. Wansing, The Logic of Information Structures. IX, 163 pages. 1993.

Vol. 689: J. Komorowski, Z. W. Raś (Eds.), Methodologies for Intelligent Systems. Proceedings, 1993. XI, 653 pages. 1993.

Vol. 695: E. P. Klement, W. Slany (Eds.), Fuzzy Logic in Artificial Intelligence. Proceedings, 1993. VIII, 192 pages. 1993.

Vol. 698: A. Voronkov (Ed.), Logic Programming and Automated Reasoning. Proceedings, 1993. XIII, 386 pages. 1993.

Vol. 699: G. W. Mineau, B. Moulin, J. F. Sowa (Eds.), Conceptual Graphs for Knowledge Representation. Proceedings, 1993. IX, 451 pages. 1993.

Vol. 723: N. Aussenac, G. Boy, B. Gaines, M. Linster, J.-G. Ganascia, Y. Kodratoff (Eds.), Knowledge Acquisition for Knowledge-Based Systems. Proceedings, 1993. XIII, 446 pages. 1993.

Vol. 727: M. Filgueiras, L. Damas (Eds.), Progress in Artificial Intelligence. Proceedings, 1993. X, 362 pages. 1993.

Vol. 728: P. Torasso (Ed.), Advances in Artificial Intelligence. Proceedings, 1993. XI, 336 pages. 1993.

Vol. 743: S. Doshita, K. Furukawa, K. P. Jantke, T. Nishida (Eds.), Algorithmic Learning Theory. Proceedings, 1992. X, 260 pages. 1993.

Vol. 744: K. P. Jantke, T. Yokomori, S. Kobayashi, E. Tomita (Eds.), Algorithmic Learning Theory. Proceedings, 1993. XI, 423 pages. 1993.

Vol. 745: V. Roberto (Ed.), Intelligent Perceptual Systems. VIII, 378 pages. 1993.

Vol. 746: A. S. Tanguiane, Artificial Perception and Music Recognition. XV, 210 pages. 1993.

Vol. 754: H. D. Pfeiffer, T. E. Nagle (Eds.), Conceptual Structures: Theory and Implementation. Proceedings, 1992. IX, 327 pages. 1993.

Vol. 764: G. Wagner, Vivid Logic. XII, 148 pages. 1994.

Vol. 766: P. R. Van Loocke, The Dynamics of Concepts. XI, 340 pages. 1994.

Vol. 770: P. Haddawy, Representing Plans Under Uncertainty. X, 129 pages. 1994.

Vol. 784: F. Bergadano, L. De Raedt (Eds.), Machine Learning: ECML-94. Proceedings, 1994. XI, 439 pages. 1994.

Vol. 795: W. A. Hunt, Jr., FM8501: A Verified Microprocessor. XIII, 333 pages. 1994.

Vol. 798: R. Dyckhoff (Ed.), Extensions of Logic Programming. Proceedings, 1993. VIII, 360 pages. 1994.

Vol. 799: M. P. Singh, Multiagent Systems: Intentions, Know-How, and Communications. XXIII, 168 pages. 1994.

Vol. 804: D. Hernández, Qualitative Representation of Spatial Knowledge. IX, 202 pages. 1994.

Vol. 808: M. Masuch, L. Pólos (Eds.), Knowledge Representation and Reasoning Under Uncertainty. VII, 237 pages. 1994.

Vol. 810: G. Lakemeyer, B. Nebel (Eds.), Foundations of Knowledge Representation and Reasoning. VIII, 355 pages. 1994.

Vol. 814: A. Bundy (Ed.), Automated Deduction — CADE-12. Proceedings, 1994. XVI, 848 pages. 1994.

Vol. 822: F. Pfenning (Ed.), Logic Programming and Automated Reasoning. Proceedings, 1994. X, 345 pages. 1994.

Vol. 827: D. M. Gabbay, H. J. Ohlbach (Eds.), Temporal Logic. Proceedings, 1994. XI, 546 pages. 1994.

Vol. 830: C. Castelfranchi, E. Werner (Eds.), Artificial Social Systems. Proceedings, 1992. XVIII, 337 pages. 1994.

Vol. 833: D. Driankov, P. W. Eklund, A. Ralescu (Eds.), Fuzzy Logic and Fuzzy Control. Proceedings, 1991. XII, 157 pages. 1994.

Vol. 835: W. M. Tepfenhart, J. P. Dick, J. F. Sowa (Eds.), Conceptual Structures: Current Practices. Proceedings, 1994. VIII, 331 pages. 1994.

Vol. 837: S. Wess, K.-D. Althoff, M. M. Richter (Eds.), Topics in Case-Based Reasoning. Proceedings, 1993. IX, 471 pages. 1994.

Vol. 838: C. MacNish, D. Pearce, L. M. Pereira (Eds.), Logics in Artificial Intelligence. Proceedings, 1994. IX, 413 pages. 1994.

Vol. 847: A. Ralescu (Ed.) Fuzzy Logic in Artificial Intelligence. Proceedings, 1993. VII, 128 pages. 1994.

Vol: 861: B. Nebel, L. Dreschler-Fischer (Eds.), KI-94: Advances in Artificial Intelligence. Proceedings, 1994. IX, 401 pages. 1994.

Vol. 862: R. C. Carrasco, J. Oncina (Eds.), Grammatical Inference and Applications. Proceedings, 1994. VIII, 290 pages. 1994.

Vol 867: L. Steels, G. Schreiber, W. Van de Velde (Eds.), A Future for Knowledge Acquisition. Proceedings, 1994. XII, 414 pages. 1994.

Vol. 869: Z. W. Raś, M. Zemankova (Eds.), Methodologies for Intelligent Systems. Proceedings, 1994. X, 613 pages. 1994.

Vol. 872: S Arikawa, K. P. Jantke (Eds.), Algorithmic Learning Theory. Proceedings, 1994. XIV, 575 pages. 1994.

Lecture Notes in Computer Science

Vol. 835: W. M. Tepfenhart, J. P. Dick, J. F. Sowa (Eds.), Conceptual Structures: Current Practices. Proceedings, 1994. VIII, 331 pages. 1994. (Subseries LNAI).

Vol. 836: B. Jonsson, J. Parrow (Eds.), CONCUR '94: Concurrency Theory. Proceedings, 1994. IX, 529 pages. 1994.

Vol. 837: S. Wess, K.-D. Althoff, M. M. Richter (Eds.), Topics in Case-Based Reasoning. Proceedings, 1993. IX, 471 pages. 1994. (Subseries LNAI).

Vol. 838: C. MacNish, D. Pearce, L. M. Pereira (Eds.), Logics in Artificial Intelligence. Proceedings, 1994. IX, 413 pages. 1994. (Subseries LNAI).

Vol. 839: Y. G. Desmedt (Ed.), Advances in Cryptology - CRYPTO '94. Proceedings, 1994. XII, 439 pages. 1994.

Vol. 840: G. Reinelt, The Traveling Salesman. VIII, 223 pages. 1994.

Vol. 841: I. Prívara, B. Rovan, P. Ružička (Eds.), Mathematical Foundations of Computer Science 1994. Proceedings, 1994. X, 628 pages. 1994.

Vol. 842: T. Kloks, Treewidth. IX, 209 pages. 1994.

Vol. 843: A. Szepietowski, Turing Machines with Sublogarithmic Space. VIII, 115 pages. 1994.

Vol. 844: M. Hermenegildo, J. Penjam (Eds.), Programming Language Implementation and Logic Programming. Proceedings, 1994. XII, 469 pages. 1994

Vol. 845: J.-P. Jouannaud (Ed.), Constraints in Computational Logics. Proceedings, 1994. VIII, 367 pages. 1994

Vol. 846: D. Sheperd, G. Blair, G. Coulson, N. Davies, F. Garcia (Eds.), Network and Operating System Support for Digital Audio and Video. Proceedings, 1993. VIII, 269 pages. 1994.

Vol. 847: A. Ralescu (Ed.) Fuzzy Logic in Artificial Intelligence. Proceedings, 1993. VII, 128 pages. 1994. (Subseries LNAI).

Vol. 848: A. R. Krommer, C. W. Ueberhuber, Numerical Integration on Advanced Computer Systems. XIII, 341 pages. 1994.

Vol. 849: R. W. Hartenstein, M. Z. Servít (Eds.), Field-Programmable Logic. Proceedings, 1994. XI, 434 pages. 1994.

Vol. 850: G. Levi, M. Rodríguez-Artalejo (Eds.), Algebraic and Logic Programming. Proceedings, 1994. VIII, 304 pages. 1994

Vol. 851: H.-J. Kugler, A. Mullery, N. Niebert (Eds.), Towards a Pan-European Telecommunication Service Infrastructure. Proceedings, 1994. XIII, 582 pages. 1994.

Vol. 853: L. Snyder, K. Bolding (Eds.), Parallel Computer Routing and Communication. Proceedings, 1994. IX, 317 pages. 1994.

Vol. 854: B. Buchberger, J. Volkert (Eds.), Parallel Processing: CONPAR 94 – VAPP VI. Proceedings, 1994. XVI, 893 pages. 1994.

Vol. 855: J. van Leeuwen (Ed.), Algorithms – ESA '94. Proceedings, 1994. X, 510 pages.1994.

Vol. 856: D. Karagiannis (Ed.), Database and Expert Systems Applications. Proceedings, 1994. XVII, 807 pages. 1994.

Vol. 857: G. Tel, P. Vitányi (Eds.), Distributed Algorithms. Proceedings, 1994. X, 370 pages. 1994.

Vol. 858: E. Bertino, S. Urban (Eds.), Object-Oriented Methodologies and Systems. Proceedings, 1994. X, 386 pages. 1994.

Vol. 859: T. F. Melham, J. Camilleri (Eds.), Higher Order Logic Theorem Proving and Its Applications. Proceedings, 1994. IX, 470 pages. 1994.

Vol. 860: W. L. Zagler, G. Busby, R. R. Wagner (Eds.), Computers for Handicapped Persons. Proceedings, 1994. XX, 625 pages. 1994.

Vol: 861: B. Nebel, L. Dreschler-Fischer (Eds.), KI-94: Advances in Artificial Intelligence. Proceedings, 1994. IX, 401 pages. 1994. (Subseries LNAI).

Vol. 862: R. C. Carrasco, J. Oncina (Eds.), Grammatical Inference and Applications. Proceedings, 1994. VIII, 290 pages. 1994. (Subseries LNAI).

Vol. 863: H. Langmaack, W.-P. de Roever, J. Vytopil (Eds.), Formal Techniques in Real-Time and Fault-Tolerant Systems. Proceedings, 1994. XIV, 787 pages. 1994.

Vol. 864: B. Le Charlier (Ed.), Static Analysis. Proceedings, 1994. XII, 465 pages. 1994.

Vol. 865: T. C. Fogarty (Ed.), Evolutionary Computing. Proceedings, 1994. XII, 332 pages. 1994.

Vol. 866: Y. Davidor, H.-P. Schwefel, R. Männer (Eds.), Parallel Problem Solving from Nature - PPSN III. Proceedings, 1994. XV, 642 pages. 1994.

Vol 867: L. Steels, G. Schreiber, W. Van de Velde (Eds.), A Future for Knowledge Acquisition. Proceedings, 1994. XII, 414 pages. 1994. (Subseries LNAI).

Vol. 868: R. Steinmetz (Ed.), Advanced Teleservices and High-Speed Communication Architectures. Proceedings, 1994. IX, 451 pages. 1994.

Vol. 869: Z. W. Raś, M. Zemankova (Eds.), Methodologies for Intelligent Systems. Proceedings, 1994. X, 613 pages. 1994. (Subseries LNAI).

Vol. 870: J. S. Greenfield, Distributed Programming Paradigms with Cryptography Applications. XI, 182 pages. 1994.

Vol. 872: S Arikawa, K. P. Jantke (Eds.), Algorithmic Learning Theory. Proceedings, 1994. XIV, 575 pages. 1994. (Subseries LNAI).